7-2011

Lean Principles, Learning, and Knowledge Work: Evidence from a Software Services Provider

Bradley Staats
University of Pennsylvania

David James Brunner

David M. Upton

Recommended Citation

Staats, B., Brunner, D. J., & Upton, D. M. (2011). Lean Principles, Learning, and Knowledge Work: Evidence from a Software Services Provider. *Journal of Operations Management, 29* (5), 376-390. http://dx.doi.org/10.1016/j.jom.2010.11.005

Lean Principles, Learning, and Knowledge Work: Evidence from a Software Services Provider

Abstract
In this paper, we examine the applicability of lean production to knowledge work by investigating the implementation of a lean production system at an Indian software services firm. We first discuss specific aspects of knowledge work—task uncertainty, process invisibility, and architectural ambiguity—that call into question the relevance of lean production in this setting. Then, combining a detailed case study and empirical analysis, we find that lean software projects perform better than non-lean software projects at the company for most performance outcomes. We document the influence of the lean initiative on internal processes and examine how the techniques affect learning by improving both problem identification and problem resolution. Finally, we extend the lean production framework by highlighting the need to (1) identify problems early in the process and (2) keep problems and solutions together in time, space, and person.

Keywords
lean production, knowledge work, learning, operations strategy, software

Disciplines
Business Administration, Management, and Operations | Finance and Financial Management

Lean Principles, Learning, and Knowledge Work: Evidence from a Software Services Provider

Bradley R. Staats*
University of North Carolina at Chapel Hill
Campus Box 3490, McColl Building
Chapel Hill, NC 27599-3490
Tel: 919.962.7343
bstaats@unc.edu

David James Brunner
27275 Byrne Park Lane
Los Altos Hills, CA 94022
Tel: 617.276.5492
dbrunner@hbs.edu

David M. Upton
Oxford University
Park End Street
Oxford OX1 1HP UK
Tel: +44 (0)1865 288800
david@upton.com

Draft: April 13, 2010
*** 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. Special thanks are due to H. Kent Bowen for his advice and encouragement over the course of this project. We would also like to thank the special issue editors, consulting editor, and reviewers whose thought-provoking comments and suggestions have significantly improved this paper. This paper benefited, as well, from the thoughts and comments of Rob Austin, Vishal Gaur, Robert Hayes, Ananth Raman, Zeynep Ton, and Noel Watson. Finally, we are grateful to the Division of Research of Harvard Business School for funding. All errors remain our own.

Lean Principles, Learning, and Knowledge Work: Evidence from a Software Services Provider

Abstract In this paper, we examine the applicability of lean production to knowledge work by investigating the implementation of a lean production system at an Indian software services firm. We first discuss specific aspects of knowledge work—task uncertainty, process invisibility, and architectural ambiguity—that call into question the relevance of lean production in this setting. Then, combining a detailed case study and empirical analysis, we find that lean software projects perform better than non-lean software projects at the company for most performance outcomes. We document the influence of the lean initiative on internal processes and examine how the techniques affect learning by improving both problem identification and problem resolution. Finally, we extend the lean production framework by highlighting the need to (1) identify problems early in the process and (2) keep problems and solutions together in time, space, and person.

Keywords: Lean production, knowledge work, learning, operations strategy, software

1. Introduction

Lean principles, exemplified by the Toyota Production System (TPS), continue to greatly interest the operations community. Many credit Toyota's sustained success to their persistent and pervasive application of these ideas to manufacturing and management systems (Hino, 2006; Liker, 2004). This thinking has motivated many manufacturing companies to imitate, either wholesale or in part, lean principles in their improvement programs. While lean production has led to improved performance in many cases (Li et al., 2005; Shah and Ward, 2007), failed implementations are common, and as Shah and Ward (2007, p. 785) note, there is significant "confusion and inconsistency" in how lean production works and how it is best implemented.

In recent years, organizations have sought to apply lean production to knowledge work (e.g., Poppendieck and Poppendieck., 2003; Schutta, 2005). While almost all work consists of manipulating both physical goods and information, work referred to as "knowledge work" primarily involves the use of information (Drucker, 1999). The utility and impact of lean production in non-manufacturing contexts remain points of contention, leaving many managers to wonder if they are merely applying inappropriate and faddish ideas while others argue that lean principles have universal applicability (C.f. Sousa and Voss, 2001). In this paper, we ask two related questions: (1) Do principles of lean production apply to knowledge work? (2) How can we extend the existing framework of lean production to a new context that differs substantially from that in which lean was developed? To answer these questions, we report our

observations and analysis of the application of lean production at Wipro Technologies, a large Indian firm competing in the global software services industry (i.e., custom software development).

In the following section we examine lean production in the context of knowledge work and identify this paper's contribution to the literature, before discussing the principles of lean production as identified in manufacturing in Section 3. Section 4 details our case study research design while Section 5 uses quantitative data from Wipro to examine the performance of lean software development projects as compared to non-lean software development projects. Projects at Wipro are the primary way that work is delivered to customers. A *non-lean* project is executed in a traditional manner, while a *lean* project is delivered using lean principles. Section 6 then qualitatively examines how Wipro's lean initiative changed the way that the firm operated. In Section 7 we discuss extensions to the lean production model while Section 8 offers concluding remarks.

2. Lean in Knowledge Work

Prior theorists note that implementations of lean production may vary across different manufacturing settings due to contextual differences (de Treville and Antonakis, 2006). Knowledge work not only has a context separate from manufacturing, but also differs fundamentally in structure, calling into question lean principles' universal applicability. Shah et al. (2008) use a services context to document how unstable and uncertain demand does not preclude the use of lean principles. At least three more differences between manufacturing and knowledge work need to be addressed, however. First, knowledge work typically takes on a character more dynamic than that generally associated with manufacturing. Once Toyota begins producing a Camry sedan, they do not try changing it into a Tacoma truck halfway down the line if the customer changes her mind. In software services, however, such a problem is common as customers often change requirements during production. This task uncertainty is not limited to customer requirements; it can also arise from the underlying technology or external environment.

Second, knowledge work processes and their connections are often invisible. While in manufacturing it may be possible to see all pieces in a process and how they fit together, this is not the case in knowledge work. Important pieces are kept in individuals' heads or represented in symbols inside

computers. While a software manager might want to view a software engineer's work in process, in many cases that is not possible until the work is complete or nearly so. Thus, process invisibility prevents problems from being identified early enough to be solved efficiently and effectively.

Finally, knowledge work is often a design task that spans high-level architecture to low-level details (Boone et al., 2008; Clark, 1985). In other words, while a manufacturing process might consist primarily of low-level exploitation (i.e., using the same knowledge and processes repeatedly), knowledge work may undertake both high- and low-level *exploration*, sometimes simultaneously (c.f. March, 1991; March and Simon, 1993). By high- and low- level exploration we are referring to where in the process architecture that exploration takes place. For example, in the context of manufacturing, while low-level exploration might involve changes at the level of an individual worker (e.g., a new sequence of steps for installing a part), high-level exploration may involve a complete redesign of the assembly process. High-level exploration is difficult since many flaws of an existing design (the output from high-level exploration) become clear only after low-level details are resolved. An architect might create an elegant high-level design; however, during low-level work (or worse, after that work is completed), a flaw in the design may be revealed. In the case of the now "Leaning" Tower of Pisa, the flaws of an inadequate design (e.g., a foundation too small for unstable ground) were not revealed until the tower's third story was added—five years into building (Encyclopædia Britannica 1989).

Prior research on lean has been premised on notions that an established architecture will not change rapidly; that task uncertainty is low, facilitating task specification; and that workers can identify problems rapidly and accurately (Spear and Bowen, 1999; Spear, 1999). The higher degrees of task uncertainty, process invisibility, and architectural ambiguity that arise in knowledge work may therefore impede or even prevent application of lean principles to knowledge work.

This paper makes four primary contributions to the existing body of research. First, we identify challenges posed by using ideas from lean production in a knowledge work setting. Lean production is premised on the specification of both outcomes and behaviors, (i.e., the work to be completed including tasks, connections, and architectures; Nidumolu and Subramani, 2003; Spear and Bowen, 1999; Spear,

1999). However, the lack of task repetition within software services obscures the degree to which tasks can be specified and then standardized. Second, we document, quantitatively, the applicability of lean principles to knowledge work. Our empirical analysis confirms that Wipro's lean software projects achieved higher, less variable performance than did a matched comparison set on most, but not all, performance outcomes. Third, we use descriptive analysis to examine how the challenges identified above were overcome. Our qualitative work illustrates how lean production improves both problem identification and problem resolution within knowledge work. The detailed specification of tasks, connections, and architectures that lean production requires (Spear and Bowen, 1999; Spear, 1999) creates opportunity to effectively deploy an iterative development model in this context. Fourth, besides offering us a chance to examine whether lean production principles apply to knowledge work, our unique context creates opportunity to extend the lean production framework.

3. Principles of Lean Production

Toyota's successful journey of more than fifty years to become the world's most profitable auto company (Taylor, 2007) is often credited to the company's manufacturing prowess resulting from the Toyota Production System (Hino, 2006; Liker, 2004). TPS was not the product of a single conceptual breakthrough. Rather, it developed gradually over many years as the accumulation of a series of small innovations (Fujimoto, 1999). Early on, Toyota leaders did not have the economies of scale enjoyed by Ford or General Motors and believed they could not attain these, so they tried to develop a system that they imagined Henry Ford might have used in their situation (Ohno, 1988).

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

Despite significant study, the field has struggled with a lack of clarity about what lean production is and what it is not (Shah and Ward 2007). Significant recent work has defined the "how" of lean production in manufacturing (MacDuffie, 1995; Narasimhan et al., 2006; Shah and Ward, 2003, 2007). Nevertheless, bundles of practices have not been identified outside of manufacturing. Therefore, we examine the Wipro implementation through the lens of prior work that has identified "lean" principles. This brief review does not survey the literature on lean production fully, but rather focuses on work done to identify its underlying principles (See Hopp and Spearman, 2004; Shah and Ward, 2007 for surveys).

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*. Ohno, the principal creator of TPS, specifies two basic criteria characterizing lean production: just in time (JIT) and autonomation. JIT is a "pull" system in which production at each step begins only when signaled for by the customer downstream. To support JIT, Ohno developed the concept of a *kanban*, with six accompanying rules. Autonomation (sometimes called *jidoka)*, captures the notion of automation including a role for operators. The goal is not to eliminate production workers, but rather to focus them on aspects of the practice valued most highly.

The book *Lean Thinking* arose from work done by the MIT International Motor Vehicle Program (IMVP). While studying all aspects of the auto industry, the IMVP focused on automobile production. In addition to the prolific output of Womack and Jones (1994, 1996, 2005; 1990), other key works associated with the IMVP include Krafcik (1988), MacDuffie (1995, 1997; 1996), and Cusumano (1998). In *Lean Thinking*, Womack and Jones (1996) narrow lean management concepts to five categories: *value*, *value stream*, *flow*, *pull*, and *perfection*. *Value* defines the use that a product offers a customer, and works backward to build the production process. Firms map production (create a *value stream*) to ensure that each step provides value. *Flow* reorganizes processes so 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 customer needs and improve one's process—with zero defects.

As noted by Hopp and Spearman, Ohno's book was "clear on basic philosophy" but "short on details," 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, p. 141). Spear and Bowen (1999) sought to resolve this discrepancy by identifying the principles of TPS inductively. Spear (1999) conducted an ethnographic study in which he visited 33 sites of Toyota and its suppliers. From this effort and analyses, Spear and Bowen (1999) derived four rules characterizing TPS:

> ***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, p. 98).

These rules capture four essential aspects of lean systems: specified tasks, streamlined communication, simple process architecture, and hypothesis-driven problem solving. Table 1 below further details the rules. In the next section, we describe the setting and methodology used in our study.

**********INSERT TABLE 1 ABOUT HERE**********

4. Case Study Methodology

4.1. Case Study Selection

In case study research, the target organization is not randomly sampled, but rather chosen according to its theoretical characteristics and how these contribute to the research question being asked (Eisenhardt, 1989; Glaser and Strauss, 1967; Siggelkow, 2007). The ideal candidate organization for this study possesses four characteristics. First, an ideal organization competes in a knowledge work industry. The knowledge work condition introduces the challenge that much of the work being completed is invisible to observers. Second, an ideal industry has high task uncertainty, which could arise from any number of sources, including environmental change or customer involvement in production that introduces the possibility of changing requirements (i.e., a customer's choices might change the work completed). Third, the architecture of work being completed should not be set. Prior work has examined lean aspects of knowledge work settings, e.g., healthcare treatment processes (Shah et al., 2008; Spear, 2005) in contexts

with a defined high-level architecture. Finally, the lean initiative should occur contemporaneously with the research study. By conducting a study as events unfold, researchers are able to view the process taking place, as opposed to just the outcome. This also helps to avoid retrospection bias (Miles and Huberman, 1994).

We identified Wipro Technologies as a research site when one member of the research team was visiting the company in January 2005 for another project. Given the uncertainty surrounding its initiative, Wipro was interested in an external review. Following email communication to discuss research parameters, the research team and representatives from Wipro spoke by phone to launch the research in April 2005. The calls were followed by six visits to Wipro's development facilities in Bangalore, India, from August 2005 to July 2007. The research project timeline is detailed in Table 2. The first author was present for all visits, while other team members participated in selective visits.

**********INSERT TABLE 2 ABOUT HERE**********

4.2. Research Context

Wipro Technologies competes in the global software services industry. The company's service offerings include application development, engineering services, IT infrastructure management, testing, and maintenance. Wipro is diversified across technologies and industries and in June 2007, Wipro had over 72,000 employees and brought in annualized revenues greater than $4 billion (Upton and Staats, 2006).

The software services industry meets the three theoretical requirements outlined above. First, companies deliver their services through well-defined software development projects, generally including a specified objective, start date, end date, and resource requirements. Software consists of knowledge work, rarely with any physical manifestation to aid engineers as they solve problems and manipulate information. Second, compared with some packaged software development processes (e.g., by Microsoft, Oracle, and SAP), Wipro customers are quite involved in development and delivery. Solutions typically are tailored to meet an individual customer's specific needs (e.g., to join two enterprise systems). This also means that high-level architecture is rarely pre-specified, so work can vary substantially from one project to the next. Even when projects' overarching goals are the same (e.g., to customize an enterprise

resource planning system), the work involved will vary based on the customer's business processes and existing information systems.

4.3. Case Study Protocol

This study followed established case study methodology for data collection and analysis (Eisenhardt, 1989; Yin, 2003). After we selected the site and gained access, we crafted our protocol, entered the field, analyzed the data, and enfolded the literature until closure was reached (see Table 3 for a summary of our approach). We collected data using multiple methods, including interviews, meeting observation, inspection of internal documents, and analysis of archival project data (Eisenhardt, 1989; Eisenhardt and Graebner, 2007). During the project, we interviewed a total of sixty-nine individuals. We developed an interview protocol (available from the authors on request) prior to entering the field. Since direct questions from prior literature were unavailable, we derived the protocol using Spear and Bowen (1999) and Spear (1999). We then shared the protocol with colleagues for feedback and enhancement.

**********INSERT TABLE 3 ABOUT HERE**********

We followed a semi-structured interview format to elicit consistent information across respondents while remaining flexible to explore each interviewee's unique perspective. Interviews lasted one to two hours, and were recorded when possible. We took extensive field notes during all interviews. To examine the lean initiative's impact on the entire organization, we spoke with people at all hierarchical levels within Wipro. The interviews included executives (e.g., chairman and business leaders), delivery managers (project managers and their managers), quality managers, project team members, and all members of the Productivity Office (the group responsible for the lean initiative). Also, we conducted multiple interviews with key respondents. For example, we interviewed the head of the Productivity Office and the executive responsible for the initiative each time we visited. We also returned to ten individuals responsible for lean projects to ask how the focal project and lean initiative had progressed. Table 4 details the number of interviews by role, per person, and by date.

**********INSERT TABLE 4 ABOUT HERE**********

In addition to interviews, we attended multiple lean project review sessions, one lean training

session, and a quarterly review session with Wipro's internal quality group. These observations helped increase our understanding of how lean was being implemented at Wipro. During and after interviews we collected supplementary materials such as written project updates and internal papers explaining lean concepts. Following each visit, we categorized data collected, based on the four principles identified in the literature review. This iteration between fieldwork and data analysis is an important part of the case study research process (Eisenhardt, 1989; Eisenhardt and Graebner, 2007). The recursive cycling between these two tasks permits researchers to empirically ground their observations and identify anomalies to advance the theory-building process. As often happens with case study-based research, while this work began as theory testing (i.e., Do the principles of lean production apply to knowledge work?), observation of anomalies led to theory building to improve the lean production framework.

Finally, in addition to the qualitative analysis, we collected and analyzed quantitative project data for both lean and non-lean projects. Given that Wipro delivers its work to customers in the form of a software project, we use a project as our unit of analysis for the analyses that follow. Project data at Wipro is compiled in an internally developed system. The system includes detailed information on both project performance (e.g., quality and delivery schedule relative to plan) and project characteristics (e.g., contract type, project size (number of hours) and complexity (lines of code)). Data is entered by project managers at regular intervals during the project (monthly and quarterly) and upon its completion. Within Wipro, the Software Engineering Process Group, which is responsible for maintaining the quality of the project database, randomly audits projects. This database permits us to evaluate the performance of lean projects at Wipro with respect to contemporaneous non-lean projects.

4.4. Implementing Lean at Wipro

In 2004, senior managers at Wipro Technologies believed the marginal benefits of its previous quality initiative had been exhausted. In the past, partly due to customer pressures for certification, Wipro had relied on public process improvement measures such as ISO 9000 and the Capability Maturity Model (CMMI). Yet, the strengths that helped Wipro replicate their systems as they grew were imitated by competitors (Winter and Szulanski, 2001). Wipro wanted to find a new approach to quality assurance that

also would provide an operations-based advantage and a barrier to imitation (Hayes and Pisano, 1996). After examining various candidate solutions from different industries, Wipro management decided to apply the principles of lean production to software services (Upton and Staats, 2006). While Wipro personnel were not the first to consider applying lean principles to software development, they found others' thinking unsuitable, as it was not sufficiently developed for a company executing hundreds of large projects (Middleton and Sutton, 2005; Poppendieck and Poppendieck., 2003).

In mid 2004, Wipro launched a pilot lean initiative to translate ideas on lean production from manufacturing for application to software services. Wipro formed a core team that spent several months reading, visiting companies practicing lean manufacturing, and discussing ideas for implementation. To begin executing the initiative, each team member sought out a software project in which to implement a lean approach to software services while continuing other work as usual. The core team deemed eight of the ten projects selected to have been successful (over 10% improvement on the prespecified metric). Buoyed by the pilot's success, Wipro rolled out the program across the firm (Upton and Staats, 2006).

Within Wipro, the five-person Productivity Office (PO) was tasked with leading the introduction of the lean initiative and made responsible for learning about lean production in manufacturing, training, monitoring progress, and sharing best practices. One PO member was assigned to support each lean project and to conduct monthly review meetings. Lean projects were closed if the project's objectives were changed substantially (e.g., if lean production ideas were going to be applied to improving testing, but the customer eliminated testing in a project), but could not be closed if they were progressing poorly.

The company defined "lean" projects loosely since no framework existed for lean software services projects. The process was therefore exploratory. For a project to be defined as lean, a team first received training (typically a four-hour session), including information on lean principles as applied to manufacturing. The team was then required to make a good faith effort to apply the lean production ideas from this training, and explicitly use and document one countermeasure from lean (e.g., visual control boards, design structure matrices, and value stream mapping, all of which are detailed later).

5. Lean Project Performance

Before we qualitatively investigate how the lean initiative changed Wipro's internal operations, it is important to identify the initiative's empirical performance. Identifying any performance improvement and associated causalities pose two distinct challenges in lean production case studies. The former point is necessary but often ignored, and regarding the latter point, an organization that has undergone a lean initiative may improve performance, but whether this change is a result of lean is not always clear. For example, other changes may be taking place contemporaneously, or improvement may result from attention being given to an area previously ignored (Roethlisberger and Dickson, 1934). Wipro presents a unique opportunity to evaluate the performance effect of a lean initiative since the initiative was rolled out on a software project-by-project basis. Therefore, opportunity exists to compare lean projects with matched non-lean projects to verify any performance difference.

We obtained detailed data from Wipro's project management system on all lean and non-lean development projects completed between January 2005 and June 2007. After eliminating projects missing data, our sample includes a total of 92 lean and 1,111 non-lean projects (during this time, a total of 772 lean projects were completed or underway. Of this total, only 92 were development projects with no missing data). We examine development projects to provide a comparable sample across projects. Wipro executes many types of projects for customers, including maintenance and testing projects. These other types of projects are dependent upon the customer's context, so general comparisons between them are difficult. 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 comparisons using KLOC as a control for complexity. In the next section we describe the variables for the study and then in the following section we conduct the analysis comparing lean and non-lean software projects.

5.1 Variables

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: $Schedule\ Deviation = \frac{(Date\ \ Delivered\ -Scheduled\ \ Date\ \ Due)}{(Scheduled\ \ Date\ \ Due-Start\ \ Date)}$. We normalize for project length, as we expect the potential of delays to increase with project length.

We also examine the lean initiative's impact on variation. To do so, we calculate truncated deviation, coding all projects less than zero as zero and all projects greater than zero at their actual value. This measure also allows us to analyze deviation, which adversely affects 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 coding system useful as a measure of performance variation.

Effort Deviation. Similar to our investigation of schedule performance, we also examine effort deviation. The effort variable captures total number of hours expended by a team on its project. The estimation process for effort is discussed in the prior section. We calculate effort deviation as follows: $Effort\ Deviation = \frac{(Actual\ \ Effort\ -Estimated\ \ Effort)}{Estimated\ \ Effort}$. 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 any 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 a project's final stage, when the customer tests Wipro's provided code against project requirements. We divide by KLOC to control for complexity. Not every project completes CAT, so our data are reduced to 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 as 1 for lean projects and 0 otherwise. The company's broad definition—training followed by a good faith effort to apply lean thinking with at least one countermeasure—should make effects more

difficult to find since such a definition might include projects that only superficially apply lean thinking. There are two ways by which it might be considered less conservative. First, there may be a Hawthorne effect, by which improvement results from increased attention rather than from process changes (Roethlisberger and Dickson, 1934). This does not seem too likely, given Wipro's continuous focus on process improvement. Both interviews and observation confirmed that lean projects at Wipro did not receive significantly more attention or resources than did non-lean projects. Also, as the number of projects increases, the likelihood of a Hawthorne effect decreases. Second, it is possible that more able project managers (PMs) choose to implement lean techniques, and so any superior performance derives from these better PMs' generally higher capabilities. 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 overseeing non-lean projects was 2.59; for PMs on lean projects, it was 2.71. A Wilcoxon rank-sum test fails to reject the null hypothesis that both samples are from the same distribution (Z statistic = -0.103, p=0.92), suggesting that similar PMs are running lean and non-lean projects, in terms of observable traits.

Total Effort, Duration, and Complexity. Since increasing effort, duration, and complexity may decrease operational performance, we control for these factors using the estimated number of project hours, calendar days required for the project, and actual kilolines of manual code written for the project.

Type of Contract. Wipro typically uses two contract structures: time and materials (T&M) and fixed-price projects (FPP). In the first case, a customer reimburses Wipro at a specified rate for the hours Wipro's project team works. In the second case, a price is negotiated up front and Wipro bears the risk of overages. We use a variable, FPP, coded as 1 for a FPP contract and 0 for a T&M contract.

Project End 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, so we include indicator variables for each SBU. Tables 5, 6, and 7 summarize the statistics for our sample.

**********Insert Tables 5, 6, and 7 about here**********

5.2. Performance Evaluation

To evaluate lean projects' performance, we examine three samples. First, we compare lean to all non-lean projects. Because such a comparison does not take into account differences in project traits, we construct two matched samples. In the first sample, we identify control projects based on six factors (traits): SBU, contract type, end year, total effort, duration, and KLOC. We match exactly on the first three traits, then select the matched project by minimizing the Euclidean distance using the formula:

$\sqrt{\sum((Effort_{lean} - Effort_i)^2 + (Duration_{lean} - Duration_i)^2 + (KLOC_{lean} - KLOC_i)^2)}$. To avoid issues with cross-sectional dependence, we permit any project to be used as a match for only one lean project. Our approach yields a match for all lean projects. KLOC is a useful control that captures the complexity of a project. However, by excluding it we are able to evaluate all lean development projects in our data, albeit with potentially less precise matches. Therefore, we construct a second sample by excluding KLOC and following the same process as above. Since all projects do not complete acceptance testing, not all matches have a quality value. We repeat the tests below, restricting the pool so all matches have quality values, and obtain similar results. Table 8 provides a sample breakdown.

**********Insert Table 8 about here**********

We use non-parametric tests, as *ex ante* we have no reason to assume normality and our sample size is comparatively small for some tests (e.g., fifty for the quality comparison with KLOC). Repeating the analyses using parametric tests yields results that are of similar statistical significance. For each of our three samples, we compare performance on schedule deviation, effort deviation, and quality. We also examine the lean initiative's impact on variation, as measured by truncated schedule and effort deviation and quality. Table 9 summarizes the results.

**********Insert Table 9 about here**********

In each test, with respect to schedule deviation, 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 (Z statistic = 2.185, $p<0.05$). For the matched samples, we use a Wilcoxon matched-pairs signed-ranks test, finding it significant for the sample without KLOC (Z statistic = -2.132, $p<0.05$), but outside conventional levels of significance for the KLOC-matched sample (Z statistic = -0.567, $p=0.57$). Examining Tables 7 and 8 reveals that the standard deviation of truncated schedule deviation is lower in lean project samples than in non-lean project samples. Levene's test for equality of variance (Levene, 1960) rejects the null hypothesis that the two groups have equal variances in all three samples (Levene's test statistic = 9.735, 11.314, and 4.648 with $p<0.01$, $p<0.01$, and $p<0.05$, for the overall sample, matched sample without KLOC, and KLOC-matched sample, respectively).

Examining effort deviation, we see that the mean of lean projects is lower than the mean for all non-lean projects, and that the mean of the paired differences for the latter two samples is negative (implying that lean projects perform better on this metric). Similar statistical tests used in the schedule deviation case show these differences to be significant for the overall sample (Z statistic = 3.334, $p<0.01$), the matched sample without KLOC (Z statistic = -2.660, $p<0.01$) and the KLOC-matched sample (Z statistic = -2.297, $p<0.05$). From Tables 7 and 8, we see that the standard deviation for truncated effort deviation is lower in the lean project samples than in the non-lean project samples. Levene's test statistic is significant for each comparison, rejecting the hypothesis that the variances are the same (Levene's test statistic = 7.849, 14.449, and 12.224 with $p<0.01$, $p<0.01$, and $p<0.01$, for the overall sample, matched sample without KLOC, and KLOC-matched sample, respectively).

Finally, with respect to quality, we see that the mean defect rate of lean projects is lower compared to all non-lean projects, and that the mean of the paired differences for the matched sample is also lower. These differences are not statistically significant in either case, however. 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. 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 (Levene's test statistic = 3.120 and -3.145 with $p<0.10$ in each case for the overall sample and the KLOC-matched sample, respectively).

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 see no significant difference in quality. Additionally, the variance of the lean project samples is lower than that for all three of the non-lean project samples. The possibility of survival bias is an important concern in comparisons such as this. Managers of poorly performing lean projects were not able to remove their lean status, however. Some lean projects were halted, but only when the customer changed the project objectives and this invalidated the intended lean objectives (e.g., the team was going to use lean ideas for testing, but the customer moved testing to a third party). While interviews and observation revealed no inappropriate project closure, we cannot completely rule out the possibility.

While empirical analyses show that the lean initiative has positively impacted projects at Wipro, the empirical effect is not seen in all project outcome variables. Given lean production's positive impact on quality in manufacturing, the lack of effect on quality in this setting is puzzling. One possibility is that an improvement in quality will take more time. For example, it may be necessary to build in learning processes after customer acceptance testing, and these processes take time to create. Second, the impact of lean may not be seen in traditional measures of quality. Quality as measured by defects may not be the key competitive performance variable going forward. Quality can be measured many ways, 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 more factors that may heighten the revealed performance impact of the lean initiative over time. First, many early projects adopted lean principles when the projects fell behind and their managers sought ways to get back on track. We discovered a number of such examples. Software engineering experience suggests that rescuing a troubled project is difficult (Brooks, 1975), but our analysis only evaluates the end state, so we would miss lean principles' help to "save" a project. A second explanation is that, in an effort to show quantifiable results quickly, many early lean projects involved only one part of the overall project. That part of the project (e.g., coding) may have experienced savings, but when incorporated into the entire project, the gain appeared much smaller. Also, some

projects were not prepared for gains in one part of the project, so gave them back in a later part. To examine how broadly lean concepts were used with projects, 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 opportunity to increase lean engagement within projects.

6. Representative Lean Projects at Wipro

Having identified an empirical performance benefit of the lean initiative, we next turn to an examination of the four principles of lean in action at Wipro. We first use a case study of one lean project to illustrate how Wipro personnel changed existing processes as they tried to map lean production ideas to their own context. We then examine each of the four lean principles in more detail.

6.1. Telecom Product

A global telecommunications firm, Telco, is a long-time Wipro customer. For several years, Wipro had been developing and supporting software for a particular product in Telco's offshore development center at Wipro. New releases came out every year, and Wipro 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. These requirements then went through a business readiness review at Telco. Wipro used these requirements to do all of the design before writing the code.

In August 2004, the project manager learned of the new lean initiative and inquired about participating. The upcoming release had a time and materials contract lasting from August until October 2005, requiring 44 new features from 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). Under this approach, work proceeds through stages sequentially: first the design is completed, then coding takes place before testing is completed. Feedback on performance is not received until the end of the process, when fixing problems is difficult and costly (Boehm, 1981). In this project, therefore, in an effort to identify problems sooner, the project manager decided to try an iterative model

whereby multiple cycles of design-build-test are rapidly completed (Beck, 1999; MacCormack et al., 2001). The iterative approach proved useful, since in the past the customer had wanted to see demonstrations during design that 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 showed actual output, which helped Telco see new possibilities that expanded the project scope and revenues (Upton and Staats, 2006).

The iterative model also led to a different scheduling approach. The project manager split desired features into six phases or iterations. In the past, work on a project was ordered based on the project manager's feel for dependencies between activities. As part of the new approach to scheduling, the PM relied on the design structure matrix (DSM) technique (Eppinger, 2001; Smith and Eppinger, 1997a, b). In DSM, a matrix is constructed with the activities or functionalities of a project as the row and corresponding column inputs. An individual places a 1 in a cell either below or above the diagonal to denote a forward dependency or a feedback loop between two activities, respectively. After the matrix is populated, a simple Excel-based tool is used to partition and then band the matrix so that tasks are ordered effectively. The project manager used the DSM tool to order features within each of the six project phases. The process highlighted unrecognized dependencies and also changed the priority order of the features (Upton and Staats, 2006). 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."

The team also used a visual control board (VCB) to highlight the status of 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, each team member indicated what percentage of the work he had completed. Previously, a project manager would assign a feature to an engineer, then check in intermittently on the individual's process over subsequent weeks. The VCB not only provided a place for the PM to receive an overall status report (something that was not possible before) and to check whether any team member had been given too much work, but it also allowed her to identify potential problems sooner and provide targeted assistance as appropriate.

Specification of documentation, standards, and testing were all ideas generated by the team's brainstorming about how to apply lean to their project. The team used Java documents to automate the creation of detailed design documents (created after high-level design is completed, they specify for the engineer feature requirements). In the past, the team relied on manual code reviews to confirm that all developers were using the client's standards for writing code. On this project, the team put the client's coding standards (e.g., rules for naming and documenting classes, functions, and variables) into the integrated development environment (a program that developers use to write software code) to confirm that developers were using the right standards and immediately notify them if they were not (Stewart and Grout, 2001). Additionally, rather than rely on inconsistently timed builds (which integrate the system or put it together for testing), the team instituted daily builds and automated the unit test cases so code was automatically checked against pre-specified rules.

Overall, the project met its deadlines. After the first project phase was completed, however, the client changed the project from Java to .Net (a development platform with a different language and structure). The team was not experienced in .Net, so underwent two weeks of training. At this time, the team also added six inexperienced engineers since 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), in this case the team smoothly incorporated the new members, in part because of the specification that they undertook due to the lean initiative. Quality for the project was better than the Wipro norm. The customer increased the scope, so project revenue increased over twenty percent. Table 10 details ways this project differed from prior projects for Telco (Upton and Staats, 2006).

**********Insert Table 10 about here**********

We now examine how the practices in Wipro's lean initiative fit within the four principles of lean production. In cases where a practice fits under multiple principles (e.g., helps streamline communications and also solve problems, as with visual control boards), we introduce the practice under the first relevant heading and then mention other principles that apply, as well as additional benefits.

6.2. Rule 1: Specified Tasks

Improvements due to task specification as part of the lean initiative seem to be more limited than improvements stemming from the other three principles, but are still present. For example, as discussed in the Telco case, one team specified and then standardized test cases and programming rules. In a separate case, another project team found themselves wasting time classifying various errors that programmers made, so they specified a system of standardized error codes for reviewers to use upon encountering problems with code. As part of the lean initiative, a third team working for an insurance client recognized that their development process could be specified and standardized. They were creating electronic forms for the client company to use in many different countries (creating the form, then linking it to the company's back-end information systems). Recognizing the potential savings as they moved from one country implementation to the next, they decided to create a standard approach for developing all forms. This task specification led to fewer defects, less rework, and improved productivity. Problems due to the lack of task specification are examined in more detail in the Discussion section below.

6.3. Rules 2 and 3: Streamlined Communication and Simple Process Architecture

Wipro adopted a number of practices within its lean initiative that fit within the lean principles relating to Rules 2 and 3. To assist in the creation of streamlined communications and simple process architectures, Wipro introduced the design structure matrix (DSM). As outlined in the Telco case above, using activities or functionalities of a project as inputs, the DSM outputs the project dependencies and suggests an ordering of tasks (see Eppinger, 2001; Smith and Eppinger, 1997a). In other words, data are structured to streamline connections and the architecture. With the DSM, a planner could identify future conflicts (through dependencies), thus helping to resolve problems with the architecture (i.e., reduce architectural ambiguity). The DSM helped to increase process visibility since, as one PM noted:

> 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 (Upton and Staats, 2006, p. 11).

A related tool developed at Wipro is the system complexity estimator (SCE), which compares the actual software architecture to the simplest possible architecture by which each module could 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. Thus, the SCE provided a test of Rule 3 by identifying the complexity of an architectural structure and then helping managers work toward a simpler one. This also aided in resolving architectural ambiguity. An additional benefit of the SCE was that it helped managers to match appropriate people to relevant work (e.g., rookies could complete simple modules while experienced personnel worked on complex modules).

As outlined in the case example of Telco, the use of visual control boards (VCB) addresses the problem of process invisibility to help streamline communications. The VCB offers team members a visual representation of their own dependencies. So, if an individual is waiting on an input or has a question about an output, she can go directly to the appropriate person. Team members initially resisted the VCB, considering it another monitoring tool and additional "report" to fill out (Adler et al., 2009). However, over time, most found it beneficial for the reason above, among others. In the past, team members had no access to the overall project plan; the VCB now provides a project summary. The VCB also aids problem solving (Rule 4), as it is a self-diagnostic input for monitoring progress toward meeting team members' goals. The VCB additionally provides opportunity to identify problems so individuals can get help sooner. Thus, it plays a role similar to that of an andon cord in manufacturing, providing a signal that performance has deviated from expectations. One quality manager explained, "It isn't the culture [at Wipro] to ask for help." Instead of waiting for major checkpoints, possible trouble spots are clarified early on so a manager can provide assistance or training.

The VCB directly benefits the PM, as well. Communication is streamlined, since the VCB creates one place to receive project status reports. In the past, the PM had to poll all team members to gauge a project's status. The VCB also enforces discipline in the PM, requiring work to be specified in greater detail (Rule 1). With smaller, more detailed increments of work, the process is made more visible through interim checkpoints. Whereas a DSM partitions a project into modules and creates an initial ordering of tasks, a VCB is a dynamic progress-tracking system for use during project execution. PMs fill out the VCB for one or two weeks at a time. Thus, as project circumstances change due to task uncertainty, so too

does the VCB. Finally, the VCB helps with problem solving (Rule 4), as it provides a simple, self-diagnostic test to assist with problem identification: the PM can see if she is under/overloading team members. If some engineers always finish their work early (or late) then the work can be better allocated.

Wipro's use of value stream mapping (VSM) highlights a technique they applied as part of the lean initiative to help create simple process architectures. Teams used VSM to trace value-adding processes and eliminate waste. In a value stream map, a team identifies each action in a process, then categorizes the actions as Value Add or Non Value Add, helping to make processes more visible. After the value stream is mapped, the team can simplify the process. For example, a team tasked with fixing defects for an existing system was meeting its service level agreements, but decided to do a value stream map of their process. An individual was assigned to track a new defect that entered the system from the customer and note all actions taken upon it. This process identified that developers often had to wait on the tester (a bottleneck), as one tester worked with ten developers. After examining the VSM, the team decided to include testing in developers' responsibilities, then streamline communication (Rule 2) between developers for testing (i.e., each engineer was assigned another engineer to test her work). The process architecture was also altered to eliminate redundant code reviews. Altogether, team productivity increased from 1.3 defects processed per engineer per week to 2.0 defects per engineer per week.

Another change that eliminated unnecessary work and resulted in simpler process architectures was the introduction of single-piece flow. Even though every Wipro project was made to order, since work would not start until a customer requested a project, the project's work might follow a batch process. In other words, if a project needed to follow steps A and B to yield one output, and the project consisted of 300 outputs of different sorts (e.g., an electronic form or a webpage), then typically projects would follow a batch process by assigning to one team 300 A's and to another team 300 B's. For example, a team was upgrading a client's website, the prior version of which used 680 Java Server Pages (JSPs). Web pages called for one or more JSPs each. Previously, one team worked on web pages while another converted JSPs. For this project, however, each web page moved through production in a single-piece flow, so JSPs were not converted until called. As a result of this architectural change, at project

completion the team found that 200 JSPs were not called on the new site and did not need to be converted (Upton and Staats, 2006).

Similarly, teams also tried applying *heijunka*, or leveling to simplify process architectures. Heijunka is a coordination approach simplifying process architecture so production proceeds at a constant rate to meet customer demand. This may eliminate both haste and slack time that can create defects and waste resources. One team, for example, was asked by a global manufacturer to develop an electronic system to track, audit, and reassign production tools worldwide. The client had a rigid, formal delivery procedure with multiple parties, and so required a month's notice to reschedule delivery. On previous projects, therefore, when teams finished early, they had been forced to wait on delivery, thus giving back any potential savings (Upton and Staats, 2006). The team restructured its process architecture, leveling tasks such as testing to spread effort over the entire time period, rather than just completing testing at a project's end. They ended up using fewer resources, overall, and also were able to propose six revenue-generating change requests to the client to fill additional time.

6.4. Rule 4: Hypothesis-Driven Problem Solving

While Spear and Bowen (1999) discuss four principles for lean, these principles can be summarized as two key ideas. Rules 1, 2, and 3 focus on *problem identification* through specification of tasks, connections (i.e., between individuals), and process architectures, respectively. Rule 4 then focuses on resolving the problems identified, through use of the scientific method. The problem-solving process involves three steps: problem definition, generation of candidate solutions, and evaluation/selection of a solution (MacDuffie, 1997; Mukherjee and Jaikumar, 1992). Rules 1, 2, and 3 address problem definition (i.e., specifying to reveal a problem), and Rule 4 generates and selects solutions. At its core, then, lean consists of repeated cycles of problem specification and problem resolution.

Spear and Bowen note that, "It is the continual response to problems that makes this seemingly rigid system so flexible and adaptable to changing circumstances (p. 98)." The challenge then, in knowledge work compared to manufacturing environments, is that much of the work being completed is invisible. Thus, if one cannot continually surface problems (to then respond to them), adaptation may not

be possible. In knowledge work, wrong working hypotheses are much harder to see, a challenge compounded by architectural ambiguity (i.e., the underlying software architecture may be infeasible, which may not be revealed until a project is near completion). To address these issues, the introduction of lean production at Wipro increased the application of iterative methods.

The question of knowing a problem exists in software is not trivial. At the end of a development process, if the software does not work, then identifying the problem is straightforward. This method of troubleshooting is expensive and inefficient, however (Boehm, 1981). Iterative methods help to identify problems early and solve them. Each iteration constitutes a well-specified hypothesis about how the system should be structured. The hypothesis can be tested either by letting customers try to use it or by trying to add new features. By contrast, in the waterfall model, the hypothesis is poorly specified (as the high-level architecture) and cannot be tested until late in the development process.

Iterations help improve problem solving for three categories of problems: customer, project-specific, and individual. Iterations help with customer learning, as they show a customer potential solutions. This helps to codify the customer's tacit knowledge as well as educate the customer about technological possibilities. An iterative model also increases project-specific learning as engineers focus on highest-value areas first and benefit from rapid feedback and low-cost experimentation (Thomke, 1998). Iterations help with project-specific learning as interdependencies are revealed sooner, with testable output generated early in the process. Team members then can use the feedback to make behavioral or architectural changes. Finally, iterations increase individual learning: In the past, weeks or months might pass between an engineer's making a mistake and its discovery. As errors are identified sooner, rework is easier and individuals can avoid repeating mistakes.

Iterative design is not new to software (Beck, 1999; Boehm, 1985) or to product development in general (Brown and Eisenhardt, 1995; Thomke and Reinersten, 1998). Within software engineering, the rise of agile methods such as Extreme Programming (XP) and Scrum have made iterative approaches possible in response to rapid changes taking place in the end-user environment (Augustine et al., 2005; Highsmith and Cockburn, 2001; Lee et al., 2006). However, as noted by Boehm (1985), iterative methods

are difficult for third-party contractors to apply "without losing accountability and control" when requirements are uncertain. Additionally, the globally distributed nature of Wipro's services runs counter to calls for developers to collocate (Highsmith and Cockburn, 2001; Ramesh et al., 2006). Finally, at the time of Wipro's lean initiative, agile methods were typically deployed with small teams, as Kent Beck, the creator of XP, notes: "Size clearly matters. You probably couldn't run an XP project with a hundred programmers. Nor fifty. Nor twenty, probably." (Beck, 1999, p. 157) To overcome these difficulties, Wipro leveraged the high degree of specification that lean requires, providing discipline that, when combined with the flexibility of iterations, resulted in a well-functioning model (Adler et al., 1999).

Another aspect of the lean initiative that improved problem solving was increased use of periodic builds and code reviews. A periodic build integrates the system for testing, while a code review involves a check of an individual's code by either a person or software. These two processes highlight team interdependencies (e.g., indirect communication or overly complicated architectures) and improve problem solving by helping to identify errors sooner. Also, a project leader's treating builds and reviews as learning opportunities might improve the psychological safety and reporting of errors on the team (Edmondson, 1999; Siemsen et al., 2009; Tucker, 2007). One 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, in a development project for a manufacturing client, one project team inspired by the Toyota idea to "go and see a problem for yourself" developed a new problem-solving technique for testing. While the offshore team in India was in charge of development, the onsite team (i.e., the Wipro team at the customer's location) did much of the testing, permitting development times of close to 24 h/day. Communicating errors between the onsite and offshore teams was difficult as the two worked at different times; the offshore team often was unable to replicate errors found onsite. To address this, the onsite team began using web video tools (WebEx) to record the exact action sequence and system build/configuration that generated the error. By using technology to reach across the temporal and

geographic separation (O'Leary and Cummings, 2007), the teams were able to see problems themselves to incorporate situated knowledge (Tyre and von Hippel, 1997).

Tables 11 and 12 provide additional detail on practices used by Wipro in its lean initiative.

**********INSERT TABLES 11 & 12 ABOUT HERE **********

7. Discussion

This study highlights a systemic issue in applying principles of lean production to knowledge work: to what degree can tasks be specified within the context of knowledge work? Lean production requires that both outcomes and behaviors (i.e., tasks) be specified (Nidumolu and Subramani, 2003). Spear and Bowen's (1999) principles identify the need to specify work at increasingly lower levels. First the architecture is specified, then the connections, and finally the work itself. Wipro's lean initiative shows great progress with respect to the first two types of behavioral specification, but relatively little on the final point. Prior work outside of lean production notes that the appropriate investment in task specificity depends on how often the task is repeated (Eisenhardt, 1985). In software services, the individual task tends not to be repetitive. Also in software services, architectural ambiguity is high (i.e., the high-level architecture is not fixed). The work involves architecting the system, not assembling systems in accordance with a known architecture. When the architecture is unknown, the interdependency structure is ambiguous and mutable, making tasks difficult to specify. With lean tools such as the visual control board, project managers are better able to specify outcomes (e.g., "this code will be completed in two days"); whether behaviors to complete the tasks can or should be specified remains an open question, however (c.f. Nidumolu and Subramani, 2003).

This study also offers opportunity to extend the underlying framework of lean production. By examining the application of lean production in a new area—knowledge work—we gain insight into the foundations that make lean successful in not only that area, but other areas as well, more generally. Spear and Bowen's four rules are premised on the idea that an organization can best identify problems in context. By studying Toyota factories at an advanced stage of lean production implementation, Spear and Bowen examine an environment with a low level of task uncertainty and a high-level process architecture

that does not change rapidly (i.e., low architectural ambiguity). In such an environment, problem identification can easily be taken for granted. In the case of knowledge work in general, and software services in particular, this process is not trivial, however. In fact, examining the evolution of production techniques at Toyota and its competitors shows that even in automobile manufacturing, identifying problems is not inherently a trivial exercise.

As Weick (1979, 1993) notes, the identification of problems is a social exercise (i.e., problems are subjective). For example, Toyota problematized waste in a way General Motors did not. Taiichi Ohno, the principal creator of the Toyota Production System, noted, "To get rid of waste, train your eyes to find waste and then think about how to get rid of the waste you've found. Do this over and over again, always, everywhere, relentlessly and unremittingly." (Hino 2006, p. 241) After learning how to identify problems, a well-trained Toyota employee can walk into another company's plant and find waste, but this characteristic is not innate. Over time, Toyota has built a system that can identify problems relatively automatically, but this took many years of experience making incremental innovations (Fujimoto, 1999). To apply lean production in a sufficiently novel setting, one would expect similarly significant effort to be necessary to learn how to identify problems in the new context. This leads the following proposition:

Proposition 1: Problems should be identified as frequently as possible, as early as possible.

As discussed earlier, lean production is a methodology encompassing various techniques that enable the continuous identification and resolution of problems. Toyota has structured the problem-solving process to maximize chances of successful problem resolution. In Rule 4, Spear and Bowen (1999) highlight the importance of using the scientific method to resolve problems. To maximize the speed and quality of problem-solving cycles, it is important not only to use the scientific method, but also to keep problems and solutions together in person, space, and time (Bowen, 2005). First, as Spear and Bowen (1999) implied in Rule 4 (recommending that problems be solved "at the lowest possible level"), the individual conducting the work where a problem occurred should address the problem. This draws on a general principle of problem solving, that knowledge about a task often comes from completing the task; therefore, collocation of the action and information enables more effective learning and

improvement (Tucker, 2004). As in the example of software code reviews, keeping the problem and solution together in a single person also can prevent an engineer from repeating a mistake.

The person is not the only dimension upon which problems and solutions should be collocated. Space (i.e., location) also plays an important role in problem solving, due to the contextual knowledge that is often embedded there. For example, Tyre and von Hippel found that in the introduction of new process equipment, some problems cannot be resolved without the development engineers' traveling to users' locations (Tyre and von Hippel, 1997; von Hippel and Tyre, 1995). We found a similar situation at Wipro in the example where the team in India uses WebEx to "travel" to the customer's location. Finally, by creating opportunity for individuals to solve problems immediately after their occurrence, for example with the use of andon cords at Toyota or frequent reviews at Wipro, the likelihood of successful resolution increases. Relevant knowledge is more accessible to the problem solver (e.g., she remembers the action steps she followed), and the opportunity to use it to identify causal linkages increases the probability that learning may occur. One project manager at Wipro noted, "When an engineer learns about a problem right after he wrote some code, he'll remember what he did to cause the mistake and he'll remember not to do it again." This leads to the following proposition:

Proposition 2: Problems and solutions should be kept together in time, space, and person.

8. Conclusion

In this paper, we see that implementation of a lean production system in knowledge work is possible and that it changes how the organization learns through hypothesis-driven problem solving, streamlined communications, simplified process architectures and, to a lesser degree, specified tasks. In its attempt to implement a lean production system, we see that core processes were altered, resulting in improved operational performance. Like any study, ours has limitations, so one should be cautious in applying its results. In particular, our study examines implementation of lean production in knowledge work, investigating 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. This limitation is a necessary but unattractive consequence of the detail and

lack of recall bias that the real-time nature of our study permits. Finally, question arises as to whether Wipro is actually doing "lean" production. Our study does not rely on the epistemological concern of whether Wipro's approach is truly lean, however. Since no definition of "lean" in software is accepted, we rely on the fact that Wipro was consciously *trying* to create a lean system for software services. Their ideas were inspired by lean thinking, in any event, so we are able to learn from their attempted mapping.

This paper makes several contributions to the existing knowledge base. First, we identify a significant challenge to using ideas from lean production in a knowledge-based industry: lack of repetition. Second, our empirical examination suggests that manufacturing-based principles are applicable to knowledge work. Third, we use descriptive analysis to examine how the challenges identified above, were overcome. Our qualitative work illustrates how lean production improves both problem identification and problem resolution within knowledge work. Finally, we extend the framework of lean production, offering two propositions for problem solving. We hope that, together with other recent work (e.g., Shah et al., 2008), our observations of details during implementation provide the beginnings of a roadmap for other knowledge-based industries seeking to apply the same ideas (C.f. Boyer et al., 2005). Such details are the most important (and most often underemphasized) part of any lean initiative, far outstripping the import of a strategic mandate that 'we are doing lean'.

While implementation of its lean production system is far from complete, Wipro's new approach may offer a way to manage uncertain and complex projects through a high-assurance, iterative model. If the greatest value from lean production comes when complementary practices are implemented as a system (Hino, 2006; MacDuffie, 1995), then this process may take some time as the company finds the right contextually dependent set of practices to eventually constitute the "Wipro Production System." While proceeding by analogy to physical processes has led Wipro's lean initiative to this point, they will need to use knowledge gained from their context to proceed further. As the company's earlier public, external approach to process management left Wipro open to imitation, its switch to lean techniques may prove beneficial if more difficult for competitors to emulate. This may, in turn, create opportunity for an ongoing operations-based competitive advantage.

9. References

Adler, P.S., Benner, M., Brunner, D.J., MacDuffie, J.P., Osono, E., Staats, B.R., Takeuchi, H., Tushman, M., Winter, S.G., 2009. Perspectives on the productivity dilemma. Journal of Operations Management 27 (2), 99-113

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

Augustine, S., Payne, B., Sencindiver, F., Woodcock, S., 2005. Agile project management: Steering from the edges. Communications of the ACM 49 (10), 48-54

Beck, K., 1999. Extreme Programming Explained: Embrace Change. Addison-Wesley, New York

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

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

Boone, T., Ganeshan, R., Hicks, R.L., 2008. Learning and knowledge depreciation in professional services. Management Science 54 (7), 1231-1236

Bowen, H.K., 2005. Personal Correspondence, Boston

Boyer, K.K., Swink, M., Rosenzweig, E.D., 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. Addison-Wesley, New York

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

Clark, K.B., 1985. The interaction of design hierarchies and market concepts in technological evolution. Research Policy 14 (5), 235-251

Clark, K.B., Fujimoto, T., 1991. Product Development Performance. HBS Press, Boston

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

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

Drucker, P.F., 1999. Management Challenges for the 21st Century, 1st ed. HarperBusiness, New York

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

Eisenhardt, K.M., 1985. Control: Organizational and economic approaches. Management Science 31 (2), 134-149

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

Eisenhardt, K.M., Graebner, M.E., 2007. Theory building from cases: Opportunities and challenges. Academy of Management Journal 50 (1), 25-32

Eppinger, S.D., 2001. Innovation at the speed of information. Harvard Business Review 79 (1), 149-158

Fujimoto, T., 1999. The Evolution of a Manufacturing System at Toyota. Oxford Univ Press, New York

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

Glaser, B.G., Strauss, A., 1967. Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Transaction, New York

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

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

Highsmith, J., Cockburn, A., 2001. Agile software development: The business of innovation. Computer 34 (9), 120-127

Hino, S., 2006. Inside the Mind of Toyota. Productivity Press, New York
Hopp, W.J., Spearman, M.L., 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
Leaning Tower of Pisa, 1989. Encyclopædia Britannica, 15th ed, Chicago, p. 223
Lee, O.-K., Banerjee, P., Lim, K.H., Kumar, K., Hillegersberg, J.v., Wei, K.K., 2006. Aligning IT components to achieve agility in globally distributed system development. Communications of the ACM 49 (10), 48-54
Levene, H., 1960. Robust tests for equality of variances, In: Olkin, I. (Ed), Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling, Stanford Univ Press, Palo Alto, CA, pp. 278-292
Li, S., Subba Rao, S., Ragu-Nathan, T.S., Ragu-Nathan, B., 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. McGraw-Hill, New York
MacCormack, A., Verganti, R., Iansiti, M., 2001. Developing products on "Internet time": The anatomy of a flexible development process. Management Science 47 (1), 133-150
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., Sethuraman, K., Fisher, M.L., 1996. Product variety and manufacturing performance: Evidence from the international automotive assembly plant study. Management Science 42 (3), 350-369
March, J.G., 1991. Exploration and exploitation in organizational learning. Organization Science 2 (1), 71-87
March, J.G., Simon, H.A., 1993. Organizations, 2nd ed. Blackwell, Cambridge, MA
Middleton, P., Sutton, J., 2005. Lean Software Strategies. Productivity Press, New York
Miles, M.B., Huberman, A.M., 1994. Qualitative Data Analysis: An Expanded Sourcebook, 2nd ed. Sage Publications, Thousand Oaks, CA
Monden, Y., 1983. Toyota Production System: Practical Approach to Management. Industrial Engineering and Management Press, Norcross, GA
Mukherjee, A.S., Jaikumar, R., 1992. Managing organizational learning: Problem solving modes used on the shop floor, INSEAD Working Paper 92/63/TM
Narasimhan, R., Swink, M., Kim, S.W., 2006. Disentangling leanness and agility: An empirical investigation. Journal of Operations Management 24 (5), 440-457
Nidumolu, S.R., Subramani, M.R., 2003. Combining process and structure approaches to managing software development. Journal of Management Information Systems 20 (3), 159-196
O'Leary, M.B., Cummings, J.N., 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. Productivity Press, Cambridge, MA.
Poppendieck, M., Poppendieck., T., 2003. Lean Software Development: An Agile Toolkit. Addison Wesley, Boston
Ramesh, B., Cao, L., Mohan, K., Xu, P., 2006. Can distributed software development be agile? Communications of the ACM 49 (10), 41-46
Roethlisberger, F.J., Dickson, W.J., 1934. Management and the Worker. Harvard University, Boston
Royce, W., 1970. Managing the development of large software systems. Proceedings, IEEE Wescon
Schutta, J.T., 2005. Business Performance Through Lean Six SIGMA: Linking the Knowledge Worker, the Twelve Pillars, and Baldrige. ASQ Quality Press, Milwaukee, WI
Shah, R., Goldstein, S.M., Unger, B.T., Henry, T.D., 2008. Explaining anomalous high performance in a health care supply chain. Decision Sciences 39 (4), 759-789

Shah, R., Ward, P.T., 2003. Lean manufacturing: Context, practice bundles, and performance. Journal of Operations Management 21 (2), 129-149
Shah, R., Ward, P.T., 2007. Defining and developing measures of lean production. Journal of Operations Management 25 (4), 785-805
Siemsen, E., Roth, A.V., Balasubramanian, S., Anand, G., 2009. The influence of psychological safety and confidence in knowledge on employee knowledge sharing. Manufacturing & Service Operations Management 11 (3), 429-447
Siggelkow, N., 2007. Persuasion with case studies. Academy of Management Journal 50 (1), 20-24
Smith, R.P., Eppinger, S.D., 1997a. Identifying controlling features of engineering design iteration. Management Science 43 (3), 276-293
Smith, R.P., Eppinger, S.D., 1997b. A Predictive Model of Sequential Iteration in Engineering Design. Management Science 43 (8), 1104-1120
Sousa, R., Voss, C.A., 2001. Quality management: Universal or context dependent? Production and Operations Management 10 (4), 383-404
Spear, S., Bowen, H.K., 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. Harvard Business School, Unpublished doctoral dissertation
Spear, S.J., 2005. Fixing health care from the inside, today. Harvard Business Review 83 (9), 78-91
Stewart, D.M., Grout, J.R., 2001. The human side of mistake-proofing. Production and Operations Management 10 (4), 440-459
Strauss, A.L., Corbin, J.M., 1990. Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Sage Publications, Newbury Park, CA
Taylor, A., III, 2007. America's best car company. Fortune 155 (5), 98-106
Thomke, S., Reinersten, D., 1998. Agile product development: Managing development flexibility in uncertain environments. California Management Review 40 (1), 8-30
Thomke, S.H., 1998. Managing experimentation in the design of new products. Management Science 44 (6), 743-762
Tucker, A.L., 2004. The impact of operational failures on hospital nurses and their patients. Journal of Operations Management 22 (2), 151
Tucker, A.L., 2007. An empirical study of system improvement by frontline employees in hospital units. Manufacturing & Service Operations Management 9 (4), 492–505
Tyre, M.J., von Hippel, E., 1997. The situated nature of adaptive learning in organizations. Organization Science 8 (1), 71-83
Upton, D.M., Staats, B.R., 2006. Lean at Wipro Technologies. Harvard Business School Case No. 9-607-032, Boston
von Hippel, E., Tyre, M.J., 1995. How learning by doing is done: problem identification in novel process equipment. Research Policy 24 (1), 1-12
Weick, K.E., 1979. The Social Psychology of Organizing, 2d ed. Addison-Wesley, Reading, MA
Weick, K.E., 1993. The collapse of sensemaking in organizations: The Mann Gulch disaster. Administrative Science Quarterly 38 (4), 628-652
Winter, S.G., Szulanski, G., 2001. Replication as strategy. Organization Science 12 (6), 730-743
Womack, J.P., Jones, D.T., 1994. From lean production to the lean enterprise. Harvard Business Review 72 (2), 93-103
Womack, J.P., Jones, D.T., 1996. Lean Thinking. Simon & Schuster, New York
Womack, J.P., Jones, D.T., 2005. Lean consumption. Harvard Business Review 83 (3), 59-70
Womack, J.P., Jones, D.T., Roos, D., 1990. The Machine That Changed the World. Rawson Associates, New York
Yin, R.K., 2003. Case Study Research: Design and Methods, 3rd ed. Sage Publications, Thousand Oaks, CA

Tables

Table 1. Four Principles of TPS (Spear and Bowen, 1999; Spear, 1999)

Rule	Explanation
1. Task Specification	Tasks are specified for two primary reasons: • First, specification 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. • Second, when work is specified and its actual condition is compared to its expected condition immediately after the work is completed, then opportunity for improvement increases. Since knowledge about a task often comes from completing the task, collocation of the action and the feedback information enables more effective learning and improvement.
2. Streamlined Communication 3. Simple Process Architecture	Coordination in a lean system is direct and simple, and can be broken into two constituent parts: connections and process architectures. A connection describes the linkage of two individuals in the system, while process architecture consists of the connections that make up the flow of goods, services, or information throughout the organization. There are three themes for coordination in TPS: • First, TPS highlights the importance of streamlining. This not only reduces complexity, but as the process which goods/services follow is streamlined, 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. With streamlined communications and simple process architectures, individuals spend less time thinking about what to do and instead work 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 (1) a hypothesis has failed and (2) he needs help. • The final theme is problem identification. Each connection or process architecture should be self-diagnostic, i.e., hypotheses are checked and revised if false.
4. Hypothesis-Driven Problem Solving	Problem solving in TPS 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. • 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.

Table 2. Timeline of research study

Date	Event
Jan 2004	Wipro begins exploring using lean in software services
Aug 2004	Wipro begins pilot projects
Jan 2005	Member of research team learns about initiative during visit to Wipro for another project
April 2005	Phone interviews to launch the project and to learn more about the lean initiative
Aug 2005	Two weeks onsite at Wipro's facilities in Bangalore
Jan 2006	Two weeks onsite at Wipro's facilities in Bangalore
July 2006	One week onsite at Wipro's facilities in Bangalore
Feb 2007	One week onsite at Wipro's facilities in Bangalore
May 2007	One week onsite at Wipro's facilities in Bangalore
July 2007	One week onsite at Wipro's facilities in Bangalore

Table 3. Detail on case study methodology (structure from Eisenhardt, 1989)

Step	Summary
1. Getting started – Definition of Research ?	Do the principles of lean production apply to knowledge work? How can the conceptual framework of lean production be improved?
2. Selecting Case	Desired case is a company implementing lean principles in a knowledge work industry. Industry should have some degree of task uncertainty and a high-level process architecture that is not fixed. Additionally, lean roll-out should be underway to avoid issues of retrospection bias.
3. Crafting Protocol	Reviewed literature on lean principles and practices and software engineering methodologies (e.g., CMMI and agile).
	Developed semi-structured interview protocol (available from authors).
4. Entering the Field	Conducted phone interviews in April 2005 with individuals from the Productivity Office to prepare for first visit.
	Visited Wipro's facilities in Bangalore six times from August 2005 to July 2007 for a total of eight weeks spent in India. During visits conducted semi-structured interviews, toured facilities, observed lean training session, observed lean project review sessions, and quarterly review session with internal quality group.
	Collected written materials such as lean project updates and lean concept notes as well as project data from Wipro's internal project management system.
5. Analyzing Data	Separated observed actions into different categories of lean principles.
	Identified anomalous actions that did not fit within existing framework.
	Conducted analysis to verify that lean projects outperformed non-lean projects.
6. Shaping Hypotheses	Marshaled data to examine why lean might improve performance in this setting
	Used anomalous observations to build theory on how to improve the conceptual framework of lean.
7. Enfolding Literature	Compared emergent findings with work not only on lean production, but also on operations management and organization theory.
8. Reaching closure	Stopped iterative analysis when we reached theoretical saturation – the point at which new evidence did not appear (Strauss and Corbin, 1990).

Table 4a. Breakdown of interviews by role

Role	# of Interviews
Executives	13
Delivery managers	29
Quality managers	14
Project team members	6
Productivity office	7
Total Individuals Interviewed	69

Table 4b. Breakdown of interviews by person

Times Interviewed	Individuals
One interviews	51
Two interviews	12
Three interviews	3
Four interviews	1
Seven interviews	2
Total Individuals Interviewed	69

Table 4c. Breakdown of interviews by visit

Visit	# of Interviews
April 2005	4
Aug 2005	42
Jan 2006	29
June 2006	5
Feb 2007	10
May 2007	6
July 2007	6
Total Interviews	102

Table 5. 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

Table 6. Correlation table for dependent, independent and control variables of interest

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 6. Breakdown of lean projects by business unit and project end year

	Business Unit		
Project End Year	**1**	**2**	**3**
2004	5	8	6
2005	14	19	14
2006	6	11	9

Table 8. 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 9. Summary results for tests of lean projects versus three comparison groups

Dependent Variable	Lean Projects Tested Against: 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 for two-tailed tests. Sample sizes are as follows: entire dataset compares 92 lean projects to all 1,111 non-lean projects (50 and 474 projects for quality); match no KLOC compares 92 matched lean and non-lean projects; match with KLOC 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.
(2) For the entire dataset comparison, we run a Wilcoxon rank-sum test, while for the latter two
(3) We calculate Levene's test statistic between each of the lean and non-lean project samples.
(4) Since quality is scaled by KLOC the measure is not used in the sample without KLOC.

Table 10. Comparison of prior non-lean Telco projects to lean Telco project

Activity	Projects before Lean	Lean Project
Project management methodology	Waterfall	Iterative
Scheduling	Project manager making intuitive decisions	Project manager using Design Structure Matrix
Tracking work	Intermittent, conversations between the project manager and team members	Visual Control Board
Detailed Design Documentation	Manually create detailed design documents	Automated the process using Java documents
Programming Standards	Manually check code to see if it conforms to standards	Coding standards placed in integrated development environment to automatically test code against standards
Testing	Manual, intermittent tests	Daily builds with pre-specified test cases

Table 11. Summary case study evidence of the four lean principles

Rule 1: Specified Tasks	Rule 2: Streamlined Communication	Rule 3: Simple Process Architecture	Rule 4: Hypothesis-Driven Problem Solving
Standardized error codes	Visual control boards	Value stream mapping	Iterations
5S	Design structure matrix	Single piece flow	Periodic builds
Pre-specified test cases	System complexity estimator	Heijunka	Periodic code reviews
	WebEx to connect engineers at different locations		WebEx to go and see a problem at another location

Table 12. Detailed case study evidence of the four lean principles (Upton and Staats, 2006)

Specified Tasks	*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.
Streamlined Communication	*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.
Simple Process Architecture	*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 process architecture 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.
Hypothesis-Driven 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 the team 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.