ESSAYS ON OPERATIONAL PRODUCTIVITY AND CUSTOMER SATISFACTION IN OFFSHORE SOFTWARE PROJECTS

SOWMYA SRIRAM NARAYANAN

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Kenan-Flagler Business School

CHAPEL HILL

2007

Approved By:

Prof. Jayashankar M. Swaminathan (Advisor)

Prof. Sridhar Balasubramanian (Advisor)

Prof. Ann E. Marucheck

Prof. Albert H. Segars

Prof. Harvey M. Wagner

ABSTRACT

SOWMYA SRIRAM NARAYANAN: Essays on Operational Productivity and Customer Satisfaction in offshore Software Projects (Under the joint supervision of Prof. Jayashankar M. Swaminathan and Prof. Sridhar Balasubramanian)

In recent times, both academia and practitioners have increasingly focused on the importance of offshore outsourcing. Analysts estimate that the offshore component of IT services is expected to rise to $70 billion by 2007. Despite this increase, the popular press has cited dissatisfaction among firms that have outsourced software projects to offshore locations. Primary reasons cited for the customer dissatisfaction with outsourcing include the increased complexity of managing the relationship, reduced productivity and reduced operational effectiveness. This issue has not received much academic attention. This dissertation attempts to address this gap in the academic literature by studying the problem from two different perspectives of a software supply chain.

The first perspective is effectiveness – where the focus is on managing the internal processes to have a positive impact on customers. This is important, because a satisfied customer is key to a successful and profitable organization. Accordingly, in Chapter 2 of this dissertation, we study the determinants of project performance and customer satisfaction in outsourced offshore software projects. The second perspective is the internal efficiency – where focus is on increasing the efficiency of processes and people; thus, leading to increase in productivity. Clearly these two perspectives are intertwined. An understanding of factors affecting productivity of individuals will enable the managers to set appropriate goals for team members, improve delivery performance, and ultimately increase customer satisfaction. Chapters 3 and 4 of this dissertation investigate productivity improvement using software maintenance as a context. In Chapter 3, we investigate the role of both individual-level factors, such as overall experience, task variety, and newness of task handled, and team-level factors such as team size, new team member entry, and team member exit, on individual productivity. Next, in Chapter 4, we investigate how productivity can be improved by better allocation of individual's effort to tasks that have the following property: the longer it takes to resolve the task, the less is the likelihood that the task will be completed successfully.

ACKNOWLEDGEMENTS

No words of gratitude are enough to express my indebtedness to my mother, father, sisters and my brother in law for supporting me throughout this Ph.D program. I feel lucky and grateful to have been benefited from the knowledge of my advisors Professors Jayashankar M. Swaminathan and Sridhar Balasubramanian for their support through the dissertation process. I will be forever indebted to them for their insights and advice. My sincere thanks and gratitude go to my committee members Prof. Ann E. Marucheck, Prof. Albert H. Segars and Prof. Harvey Wagner for their valuable time, commitment, support and feedback during the dissertation process. I also want to thank Dr. Chariyath Suchindran of the UNC Biostatistics Department for his time in discussing details of the fourth chapter of this dissertation. I also want to thank the employees at the firm we worked with, and their exceptional patience and support in working with me through the teleconference calls during the course of the data gathering process. I must also commend their patience in taking the time to sift through several details that I needed and clarifications they provided during the course of the dissertation writing. Finally, I would also like to thank the consultants at the Odum Institute at the UNC – specifically, Chris Weisen, Sharon Christ and Daniel Serrano – who all were ready to lend a helpful ear and discuss some of my results and problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

1. Introduction

1.1. Outsourcing overview and research motivation

An emerging global workforce, greater availability of information technology, and increasing global competition are redefining service delivery value chains by enabling the disaggregation of services (Quinn 1992). The ability to disaggregate activities in products and services has led to increased outsourcing of products and services (Apte and Mason 1995). IT services, including software based services, have been a part of this phenomenon. The scale of outsourcing in software services can be judged from the fact that the International Data Corporation (IDC) estimated the global demand of software-based IT services to be USD 382.1 billion, of which, outsourced IT services were USD 118.2 billion (IDC 2003). Much of the outsourced work is 'offshored' and goes to developing nations in various parts of the world like India and China. The resulting global supply chain of products and services presents challenges to managers in coordinating and executing work (Swaminathan and Tayur 2003).

For supply chain partners, these challenges can lead to problems in delivering expected output. In the outsourced software services domain, evidence of such problems are starting to appear. For example, a recent practitioner survey reported that the overall satisfaction index of outsourcers fell to 6.4 on a scale of 10 in 2004, as compared to 7.1 in 2003. The study also found that only 62% of the respondents were satisfied with their outsourcing partners in 2004 as compared to 79% a year ago in 2003. In particular, offshored projects were cited as difficult to manage because of the complexity in managing the relationship, and problems of cultural adjustment between teams working in different countries (Mcdougall 2004). Another study found an appalling 51% of the clients wanting to terminate their outsourcing contracts due to lack of satisfaction with outsourced software projects, citing poor quality, reduced operational effectiveness and greater management complexity as the primary reasons (McEachern 2005). These studies also indicate that the problems lie with both the service provider and clients alike.

From a customers' perspective, dissatisfaction could be due to systemic problems with managing offshore processes, having wrong or unreasonable expectations, and lack of awareness of how to make offshore outsourcing to succeed. For example, Gartner – a leading market research agency – found that one of the top 5 reasons for failure of offshore projects was the general tendency of firms to rush offshore, and enter into deals too hastily (Huntley 2005). The key reasons for failure of offshore projects were unrealized cost savings, loss of productivity, poor commitment and communications, cultural differences, lack of expertise and organizational readiness. Interestingly, the report also indicates that exploiting potential productivity advantages from offshoring needs planning and patience (Huntley 2005).

From a service provider's perspective, problems may lie in developing effective delivery capabilities such as effective project management, communication, and improving productivity of employees. These capabilities may have a significant impact on customer satisfaction. Further, developing these capabilities can be challenging given the competitive environment which is characterized by high employee turnover, shortage of talent, and demanding customers.

Despite considerable amount of research in the area of outsourcing[1], there is less understanding about what drives outsourced offshore projects to succeed, and even less so on how offshore firms can operate more efficiently. While numerous articles in the popular press have been published to educate practicing managers on issues related to offshore software outsourcing, the empirical research on the topic is limited but for a few notable exceptions (e.g. Gopal et al. 2003; Ethiraj et al. 2005). Second, there has been little work that has examined outsourcing from a service operations lens. Such a perspective can (1) help incorporate customers' evaluations of performance into better management of processes (2) lead to a better understanding of the processes in the firm and enable efficient management of activities. Integrating these perspectives can help managers and researchers better understand and manage software operations. Given that labor is one of the critical components of the overall software development cost (e.g. more than 75% of the software development costs are expended on labor costs (Amoriebata et al. 2001)), improving labor productivity is a critical problem and central to the operations in software environments. Further, improving labor productivity, in turn, may improve customer satisfaction. This dissertation aims to address these issues in software services environments and contribute to a better understanding of these dual perspectives – improving customer satisfaction and improving internal productivity – of the software supply chain. The context of this dissertation is an offshore services firm from India. The choice of an Indian firm to study management of offshored IT services is reasonable given that India, is by far, the largest destination for offshored software services (Carmel and Tija 2005). The next section outlines an overview of the Indian software services industry.

[1] A comprehensive overview of the current literature on IT outsourcing can be found in Dibbern et al. (2004)

1.2. Indian software services Industry: Overview

India has emerged to be one of the important locations in the software offshoring industry competing with several other countries such as China, Philippines, Vietnam and Russia (MacFarlan 1995). Recently India took the top spot as the choice among US businesses that are looking to outsource technology work offshore, in a survey conducted by a leading IT outsourcing consulting firm – NEO IT. Much of the initial IT services revenues for most large companies in India came from the Y2K times – year 2000 bug – when several US firms frantically sought external resources to help in achieving Y2K compliance on time. Under such conditions, many of the firms that employed Indian engineers found value in the low cost labor that they provided. Further, the quality focus of the Indian firms enabled them to expand from regular non-critical activities to executing more critical set of activities from offshore. An evidence of this can be seen from the fact that, as of December 2005, over 400 Indian companies had acquired quality certifications. Out of the 400, 85 companies were assessed at SEI-CMM Level 5. This is the highest level of process maturity that a firm can achieve. This number is the highest amongst any country in the world.

Further, in terms of variety of services, the IT services market in India has grown from providing custom application development and maintenance services to providing other services like packaged software implementation, systems integration, network consulting and integration, IT consulting, and IT support and training (NASSCOM 2005). Of all the services, the biggest share of IT services came from custom application development and maintenance which accounted for 51% of India's total exports – accounting for the largest percentage of India's exports – in the financial year 2004. These services have a very high offshore content of 85% (NASSCOM 2005). In terms of depth, export of global R&D

services, and product development services from India is expected to grow rapidly from a current USD 2.3 billion to 8-11 billion between 2008 and 2010 (NASSCOM 2005). Not only is the complexity of work managed by the Indian companies growing, the depth of the relationship in the engagements between the Indian Service providers and their clients has also been on the rise. Reflecting this trend in the increasing depth of relationship, the National Association of Indian Software and Service Companies (NASSCOM), in its strategic review of the Indian IT services industry for 2005 states that "*Over time off-shoring software has grown from one off, project based engagements involved in low end activities to longer term engagements often involving multiple, more complex tasks*" (NASSCOM 2005, p. 23). Lastly, the client base for the Indian IT-ITES firms includes a majority of the top 2000 Global corporations (NASSCOM 2005). Thus, India constitutes a very important element of global offshore environment.

1.3. Dissertation overview

To understand the dynamics of software delivery better, this dissertation divides the software supply chain into two perspectives. The first is the external perspective – where managers need to be concerned with "How the management of practices within the firm impacts their customers?" This question is important, because a satisfied customer is the key to a successful and profitable organization. The second perspective is the internally focused – where managers need to understand the antecedents to improving operational performance. This, in turn, may lead to higher customer satisfaction. An understanding of factors affecting team productivity will enable the managers to set appropriate goals for team members, improve delivery performance, and ultimately, increase customer satisfaction. This dissertation studies these two facets of a software supply chain in a sequence of three essays. The first essay investigates the determinants of customer satisfaction, and the other two essays investigate the determinants of productivity in software services.

External Perspective

This part of the dissertation focuses on the role of internal processes in improving customer satisfaction and project performance. In this essay – essay 1 – titled "*Managing Outsourced Offshore Projects: Antecedents of Project Performance and Customer Satisfaction*", we adopt a multidisciplinary perspective to investigate the antecedents of customer satisfaction and project performance in offshore projects. This work offers managerial and theoretical insights on successful management of offshore projects through an empirical analysis of real life data.

In this work, we synthesize extant literature on service operations, marketing, and software engineering disciplines to develop a conceptual model of the antecedents of project performance and customer satisfaction in outsourced software projects. Using archival survey data on customer satisfaction and project performance ratings of projects executed offshore by an Indian software services firm, we examine the influence of communication effectiveness (defined as how well the offshore team can communicate), team stability (defined as the overall longevity of engineers in the team and smoothness of work transition in case of turnover), project management capability (defined as the ability to plan work, manage priorities and handle project risks) and project performance (defined as adherence to service level agreements) on overall customer satisfaction. Further, this work also offers insights on how the impact of the antecedents on project performance and customer satisfaction vary based on the duration or length of time that the project has stayed offshore, the type of project (e.g. development and maintenance projects or testing projects), and the class of software work (e.g. application development and maintenance or system software development and maintenance). This work uses structural equations modeling to test our hypotheses.

We find that first, project management directly impacts customer satisfaction in projects that involve development and maintenance as compared to testing. Second, team stability is an important determinant of communication, project management, customer satisfaction and project performance. The role of team stability varies across the nature of the projects, and the context in which the project is considered. For example, we find that when projects have stayed offshore longer, team stability directly impacts customer satisfaction. However, when projects are relatively new, team stability directly impacts project performance, but not customer satisfaction. However, in both cases, the indirect effect is significant. Understanding these nuances can help manager's better plan activities better, manage tasks more effectively and improve customer perceptions of offshore operations. One of the key messages of this research, apart from offering several insights to manage projects better is that a one size fits all approach to managing offshore projects does not work in satisfying customers, but managers need to consider the role of antecedents based on project type, project class and project duration to get better results.

Internal perspective

This perspective of the study addresses how managers can improve the productivity of individuals using two different strategies using software maintenance as the context.

In Chapter 3 of this dissertation titled "*Individual Learning and Productivity in a Software Maintenance Environment: An Empirical Analysis*," we study the factors that drive learning and productivity in a software debugging environment. Using a panel dataset of engineers involved in performing debugging operations, we perform an empirical analysis of the determinants of learning and productivity to understand drivers of individual learning in the context of software debugging. Specifically, we investigate the role of individual level factors such as overall experience, task variety, novelty of task handled and team level factors such as team size, new hire and employee departures on individual productivity. Our analysis suggests that individuals learn from handling a variety of tasks. We also investigate the idea that individuals need a balance in exposure to task variety and task specialization. To operationalize this idea, we adapt the Herfindahl-Hirschman Index from the anti-trust literature in economics and use it as a measure of experience concentration. We investigate the role of collocation in teams and its impact on individual productivity in problem solving tasks. The findings in this essay will not only make theoretical contributions to the organizational learning and operations management literature, but they will also have implications for managers handling software projects – who often face situations of assigning individuals to multiple tasks. From a software manager's perspective, this study offers considerable insights into managing task allocation within a team and understanding the implications of task allocation.

In chapter 4 of this dissertation titled "*Optimal resource allocation in Software Maintenance,*" we investigate how the use of precious engineering resources can be optimized in a software debugging environment. This essay attempts to integrate econometric estimation procedures with traditional optimization methods in an attempt to provide managers insights on planning capacity of teams and understanding the tradeoffs between the number of tasks waiting and team capacity. We show that managers can reduce wasted effort when bugs seemingly take a long period of time by closing or providing workarounds – and reinvest the time to resolve bugs that are more likely to get resolved. We motivate such actions by empirically demonstrating heterogeneity in the bug population and verifying that the probability of resolution reduces with time, and by showing that the loss in cumulative probability of resolution is minimal as compared to the gain in the reduction of overall waiting times in queue. We also estimate a heuristic measure to determine the capacity of teams in this environment.

Data for this dissertation was obtained from a large Indian software services firm. The firm operates in more than 10 countries worldwide, employs more than 10,000 people and has upwards of $500 million in annual revenue. Most of the firms' revenue is derived from the export of software services. A field visit to the sites of the firm was conducted in summer of 2005 preceding the data collection. Data for Chapter 2 comes from archival customer satisfaction survey data consisting of 677 projects. The overall dataset for chapters 3 and 4 was drawn from debugging tasks performed by engineers in the context of software maintenance. More detail on the nature of the data used is described in the individual chapters.

CHAPTER 2

2. Managing Offshore Software Projects: Antecedents of Project Performance and Customer Satisfaction

2.1. Introduction

The last two decades have witnessed rapid advances in communication technologies, including the emergence of the Internet. These advances have enabled firms to leverage the variations in workforce skill sets, time, and cost structures across countries to create highly competitive global value chains (Swaminathan and Tayur 2003). As a result, outsourcing and offshoring of business processes has been a rising phenomenon over the last decade (Engardio 2006). The outsourcing of software work to offshore locations has been an integral part of this phenomenon. The estimated global demand for software based IT services was USD 382.1 billion in 2003, of which outsourced IT services comprised USD 118.2 billion (NASSCOM 2005). Increasingly, such outsourced activities are executed in offshore locations such as China and India. While the volume of such outsourcing has steadily increased, recent surveys suggest some dissatisfaction with vendor performance (McDougall 2004; McEachern 2005). According to McEachen (2005): "*Fifty-one percent of respondents reported terminating an outsourcing contract. On the satisfaction side, 62 percent of respondents said they were satisfied with their outsourcing relationships, down from 79 percent a year ago*." This statement highlights the need to examine the drivers of project performance and customer satisfaction in offshore projects.

Against this backdrop, our work offers the following contributions. First, to the best of our knowledge, this is the first study that examines the determinants of customer satisfaction and project performance in the offshore outsourcing context. Knowledge about the key drivers of customer satisfaction is crucial towards facilitating the appropriate allocation of resources in the management of offshore software projects. In the early stages of outsourcing, offshore service providers have focused primarily on issues related to technical performance, with only a secondary focus on project management. However, as the competitiveness of the outsourcing environment has steadily increased, these firms are now compelled to become more customer-focused. In that context, our work provides timely insights into what customers who outsource software projects are looking for, and how software service providers can deliver on those requirements.

Second, while there is substantial work across disciplines on project management in general, little is known about how the internal operational variables involved in executing software projects work in tandem to drive the customer's perceptions of project performance, and ultimately, customer satisfaction. In that sense, our work serves as a bridge that integrates insights from multiple disciplines including service operations, marketing, IT, and software engineering. We present and empirically test a model that links key antecedent variables under the control of the offshore project manager, such as team stability, communication effectiveness, and project management, and predicts their impact on the customer's evaluations of project performance and overall customer satisfaction. We distinguish between the direct and the indirect effects of these antecedents and investigate how project-related contextual factors moderate these effects.

Despite a considerable amount of work on outsourcing in general (See Dibbern et al. (2004)), there is little empirical work on the determinants of performance outcomes in outsourcing. Lee and Kim (1999) examined the determinants of partnership quality and outsourcing success. They investigated strategic dimensions of outsourcing success such as participation, joint action between the teams and top management support. Lee et al. (2004) investigate the form of outsourcing strategy – the degree of outsourcing, allocation of control and performance period – on the measured dimensions of outsourcing success. However, these investigations look at performance outcomes in achieving economic, strategic and technological benefits from outsourcing (Lee et al. 2004; Lee and Kim 1999; Grover et al. 1996). These papers do not investigate tactical issues of what determines project success in outsourced environments, but investigate relationship level determinants of outsourcing success. From a project execution perspective, the current empirical literature in the context of project execution in offshore outsourcing investigates issues that include capability building in outsourced projects (Ethiraj et al. 2005), determinants of contracting decisions in software projects (Gopal and Sivaramakrishnan 2005; Gopal et al. 2003), coordination in outsourced software projects (Gopal et al. 2002; Sabherwal 2003), and investigating the role of communications and processes in offshore software development (Gopal et al. 2002).

To summarize, there are two broad gaps in the literature on software outsourcing. First, while there have been numerous conceptual discussions and press articles on software outsourcing, the empirical literature on the performance determinants at the tactical level of outsourcing – in particular offshore outsourcing, is limited but for a few notable exceptions (e.g., Lee and Kim 1999; Lee et al. 2004; Ethiraj et al. 2005, Gopal et al. 2002, Gopal et al. 2003). Second, there has been little work that has examined offshore outsourcing from an interdisciplinary, service management perspective that incorporates customers' performance evaluations (e.g. Grover et al. 1996). Such a perspective is particularly useful in the outsourced software services context. For example, providing software services involves customization (Schmenner 1986; Stewart 2003), project management, and workforce allocation and management (Cook et al. 1999; Schmenner 1986) – this calls for insights from a service operations perspective. In addition, such services involve issues related to the design and development, testing, and ongoing maintenance of software products (Boehm 1989) – this calls for insights from IT and software engineering perspectives. Finally, such services also involve frequent and deep customer involvement which introduces goal and outcome uncertainty (Chase 1978; Larsson and Bowen 1989), the careful management of customer expectations through ongoing communications (Berry et al. 1985) and the management of customer satisfaction – this calls for insights from a marketing perspective.

In §2, we develop the conceptual model, review relevant literature and outline our hypotheses. In §3, we describe the research design and validate the measurement model. In §4, we present the results. We discuss managerial implications and ideas for future research in §5.

2.2. Conceptual Model and Hypotheses

Figure 1 describes the conceptual model that underpins our analysis. Our hypotheses are summarized as follows: Better project performance leads to higher customer satisfaction (H1). Better project management leads to better project performance (H2) and higher customer satisfaction (H3). Better communication effectiveness leads to better project management (H4). Project management mediates the impact of communication effectiveness on project performance (H5). Better communication leads to higher customer satisfaction (H6). Higher team stability has a positive impact on project performance (H7), customer satisfaction (H8), communication effectiveness (H9) and project management (H10). Project size has a nonlinear impact on project management (H11) and effective communication (H12). We use communication ability and communication intensity as indicators of effective communication. We now discuss the individual constructs and motivate the relationships outlined in the model.

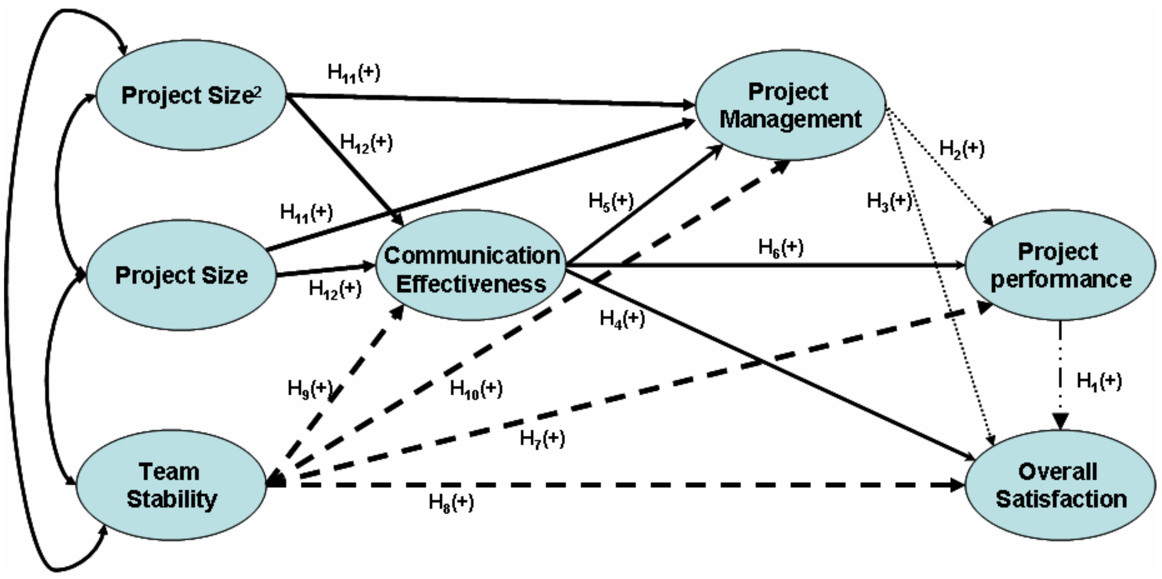

Figure 1 # Conceptual Model and Hypothesized relationships

Customer Satisfaction

Customer satisfaction is the evaluative response of the customer to the services rendered by the provider (Wirtz and Bateson 1999). Higher customer satisfaction can lead to higher firm profits (Bolton 1998). However, satisfying customers is a challenging task for firms that provide custom software engineering services. Such services involve a high customer involvement in the service delivery process, high work complexity, uncertain outcomes and long service contact durations. In the specific context of offshore services, the physical and cultural distance between the customer and the service provider can complicate the relationship between them. As noted by Stewart (2003), services are less likely to be successful when they involve intensive customization, complex tasks, remote performance and delivery (few tangibles), and contract workers who can be influenced only to a limited extent. Thus, understanding what drives customer satisfaction and managing those drivers is particularly important in the context of outsourced software services.

Satisfaction has been modeled as a function of the gap between expectations and performance (Berry et al. 1988). However, in software services, expectations may be unclear and customer satisfaction may be based on subjective customer experiences with the delivered services (Grover et al. 1996). Direct, customer experience-based measures of satisfaction are appropriate in such scenarios (Rust et al. 1999). Such measures have been adopted in earlier customer satisfaction studies (Grover et al. 1996; Balasubramanian et al. 2003; Krishnan and Ramaswamy 1999).

Project Performance

Consistent with the existing literature in software engineering (Deephouse et al. 1996; Nidumolu 1995), project performance measures output timeliness, output quality, and effective management of interim goals. Output timeliness and quality are both important components of the service-level agreements negotiated between the parties. Better product delivery can increase customer satisfaction in the service context (Bolton and Lemon 1999). Specifically in technology-intensive contexts, Krishnan and Ramaswamy (1999) found that the quality of financial statements and services provided in the banking industry were positively linked to customer satisfaction. Likewise, Balasubramanian et al. (2003) found that the operational competence of an online broker was associated with increased customer satisfaction. Building on these arguments:

H1: Project performance positively influences customer satisfaction.

Project Management

Project management in the outsourced service context covers the ability to plan and estimate work (Ethiraj et al. 2005), effectively demarcate and manage project priorities (Wallace et al. 2004), and manage project risks (Boehm 1989; Pressman 2005). Effective project management is crucial in the context of outsourced software development, because of the physical and cultural distance between customer and vendor. This distance implies that the mutual understanding and operational processes have to be in place to ensure that the project objectives are properly demarcated and those objectives are delivered in a timely manner.

First, to scope out the project, the vendor must be able to plan and estimate the total inputs that are required for the project. Careful estimation is challenging because high customer involvement usually introduces significant uncertainty in tasks and goals (Larsson and Bowen 1989).

Second, once the project is scoped out, demarcating and managing priorities, and resource allocation on an ongoing basis becomes important. Since software is an amorphous offering, there are many initiatives that managers and engineers can undertake at any point in time to make the offering even more function-rich, leading to "feature creep" (Brooks 1995).

Further, continuous customer interactions and shifting customer preferences can lead to substantial uncertainty in the scope of the project. Correspondingly, managers must be capable of scheduling resource allocations at short notice and shifting allocations of resources across time to deal with shifting project scope (Slocum and Sims 1980).

Finally, the intangibility and continuous evolution of software products increase the risks associated with successful design and development (see Barki et al. 1993 for a summary). Software development involves a high degree of risk for three reasons: First, unlike manufacturing, each software project is one-of-a-kind, with no standard (material) prototype that can be tweaked. Second, changes in the external user environment imply that software designed to fit the user's needs at some previous point in time will rarely fit perfectly with the user's needs at the time it is released. Third, the gap between the technical knowledge of the product designer and the domain knowledge of the user is particularly large in the context of software. For example, a programmer may be expert at building a standard billing system, but the billing system requirements may vary sharply across a hotel, an airline, and an online auction site.

Given these challenges, the lack of sound project management in software projects has often led to projects that "came in years behind schedule, exceeded their budget by millions, and failed to meet their users' needs even if completed eventually"(Nidumolu 1995, p.192). In contrast, good project management skills involve effective work planning and estimation, task prioritization, and risk management can improve perceived project performance (Nidumolu 1995). Building on these arguments:

H2: Project management positively influences project performance.

When customers evaluate services, they evaluate not just the final outcome but the series of encounters that lead up to that outcome – in fact, the process may play a greater role than the final outcome in determining overall customer satisfaction (Brown and Swartz 1989; Danaher and Mattsson 1994). Interaction with customers during the ongoing service delivery process is particularly important in the context of outsourced software development because the priorities, requirements, and the emerging issues in a project are often unclear to both the customer and the service provider. Good project management practices help in resolving goal and process uncertainties by getting customers on board early, and keeping them involved in the project at various stages (Stewart 2003). The effort that customers may invest in interacting with the service provider and guiding the development process can increase customer satisfaction on account of self-attribution effects (Konana and Balasubramanian 2005) and on account of customer reassurance that the service provider is sensitive to and accommodating their evolving needs (Youngdahl and Kellogg 1997).

Further, for most tangible products, the management of the development process is invisible from the customer's perspective. However, in the software services context, the customer who continuously interacts with the service provider is able to observe the implementation of behavioral and outcome controls that are part of good project management (Nidumolu and Subramani 2003). Further, the positive implications of effective work planning and estimation, task prioritization, and risk management initiatives employed by the service provider are transparent to such a customer. Building on these arguments:

H3: Project management positively influences customer satisfaction.

Communication Effectiveness

In the offshore software development context, effective communications are particularly important on account of the lack of frequent face-to-face encounters, time zone differences, and the cultural divides between the involved parties (Wright 2005). Poor communication has often frustrated managers involved with offshore services, as illustrated by the following practitioner quote:

"*…communication among the project team members is more difficult, more time-consuming, and therefore more costly. Conference calls take longer, information is misunderstood, and email volume increases. Even worse, poor communication is one aspect of any outsourcing relationship that will doom it to failure*" (Clifton 2005).

The purpose of effective communication is to keep the customer informed about project activities. Effective communication in the offshore context involves two distinct facets. First, the service provider has to provide frequent, timely, and complete reports of project progress to the customer. Second, and more generally, in the absence of face-to-face contact, the service provider has to effectively articulate issues, many of which are technically complex and prone to multiple interpretations, through oral and written communication. Accordingly, to capture communication effectiveness, we use two second order constructs to separately capture project reporting performance ("communication intensity") and the ability of the offshore engineers to communicate ("communication ability"). First, communication intensity measures the frequency and quality of project status reports sent to the customer. In the existing literature, researchers have used proxy measures to capture parallel constructs.[2] Second, communication ability measures the ability of the offshore counterparts to articulate issues when interacting through conference calls or emails. Such oral and written communication skills have been recognized as pivotal inputs into effective software project management (Curtis et al. 1988; Nidumolu 1995), particularly in the context of offshore operations (Apte et al. 1997).

Effective communications can affect multiple variables in our model. At one level, effective communications can enhance customer perceptions of good project management (Boehm 1981; Gopal et al. 2002). Here, first, effective communications can help in better work planning and estimation. Specifically, by communicating frequently and effectively with the customer, the software developer can obtain a clear idea about the needs of the customer and about the level, kind, and temporal scheduling of resources that would be required to meet those needs. Second, frequent and effective communications help the developer keep track of the shifting priorities of customers, and help align development resources to be responsive to the short-term requirements of the customers. This is particularly important in the software development context, where day-to-day fire fighting to address unexpected software bugs and implementation problems have to be coordinated with longer term development and enhancement work. Third, frequent and effective communications can help keep the gap between the current state of the project and the customer's desired project trajectory low. Customers who are constantly informed about surprises and advances in the software development process can provide quick feedback and useful inputs into further stages of development. Building on these arguments:

H4: Communication effectiveness positively influences project management.

---

[2] Other variables that reflect the degree of project-related coordination and communication include the total number of onsite customer employees the software development team has to interact with, and the frequency of

Better communications anchors superior project management, which, in turn can affect project performance (i.e., the quality and timeliness of delivered products, as seen from the customer's perspective). Nidumolu (1995) finds that the effect of communication on project performance is mediated by project risk, which in our model is controlled by superior project management. Likewise, in the financial services industry, Lievens and Moenaert (2000) find that uncertainty reduction mediates the impact of communication on project success. Building on these arguments:

H5: Communication effectiveness positively influences project performance, and this influence is mediated by project management.

Finally, communication can directly impact customer satisfaction in two ways. First, effective communications can influence how the customer evaluates different aspects of the software development process and can help in the ongoing management of customer expectations. This can increase the customer's satisfaction with the procedures involved in software development. Effective communications can also influence how the customer evaluates the final delivered product, so that the positive aspects are highlighted and the negative aspects muted. Correspondingly, effective communications can enhance the overall quality of delivered services as perceived by the customer (Berry et al. 1985; Garvin 1988; Lengnick-Hall 1996)

Second, effective communications can reduce task uncertainty (Sitkin et al. 1994). Software is ultimately a "credence" good whose properties must be discovered through post-implementation experience with the finished product. However, effective communication during development can create confidence in the customer that the vendor is on track with

---

meetings between the customer and service provider (e.g., Gopal et al. 2002, Deephouse et al. 1996).

development and that the customer's needs are being recognized and accommodated in the software design. The resulting feeling of reassurance is extremely important in a context where the development activity is taking place in a distant and unfamiliar location over which the customer has no day-to-day control. Building on these arguments:

H6: Communication effectiveness positively influences customer satisfaction.

Team Stability

This construct measures perceptions of the length of time software engineers stay in the project, and the effectiveness of the offshore team in managing work without disruption in case of turnover. Engineer turnover has been a specific area of concern in offshore software projects (Gopal et al. 2003). Turnover in a project can happen either on account of resignation or reassignment to another project. Resignations are common in a booming industry where software service providers routinely poach each other's talent. Intra-firm transfers happen because offshore software service providers often reassign experienced engineers to guide startup projects or to manage other, new high-profile projects that the firm has taken on. Alternatively, engineers who are bored with a project or seek a change in work profile to widen their portfolio of skills may themselves seek to be reassigned. In a survey of 104 Indian software firms that was conducted even at an early phase of the outsourcing phenomenon, 89 firms listed the shortage of skilled labor and 71 firms listed employee turnover as one of top three business problems they face (Arora et al. 1999).

Loss of experienced employees leads to reduction in productivity and decrease in project performance (Abdel-Hamed 1989; Oliva and Sterman 2001). Further, the long lead times associated with hiring new engineers, training them and bringing them up to speed on the customer's project compounds the problems related to maintaining productivity and work output quality. Building on these arguments:

H7: Team stability positively influences project performance.

Apart from the pursuit of cost savings, the high demand for and turnover of IT personnel within the U.S. has been cited as a key driver of offshore outsourcing initiatives (Carmel and Agarwal 2000). Consequently, customer firms that have outsourced software development and maintenance have been particularly sensitive to turnover in the ranks of the service providers themselves. These firms have often faced the situation where they pay for an engineer to learn about their systems and software, only to have that engineer leave the service provider in search of more promising opportunities soon after the "breaking in" period is completed (Overby 2003).

In the light of these negative consequences, reducing and managing turnover to minimize productivity and information losses is a priority for the service provider. High turnover can displease customers particularly when key contact people at the service provider leave – in fact, the relationship between the customer and selected employees who work for the service provider may at times be stronger than the relationship between the customer and service provider at an institutional level (Czepiel 1990; Gwinner et al. 2005).[3] While customers often prefer to deal with specific employees, customer satisfaction may be maintained by ensuring that the project team itself offers a stable set of competencies and knowledge to the customer. In that case, the negative implications of the transition of a single employee or a few employees are minimized. Initiatives that can help in this context include

[3] According to Tax and Brown (1998) American Express estimated that 30% of the customers of a typical financial advisor within the company would move with him or her to a competing firm.

the rotation of key employees, use of teams that comprise individuals with overlapping skill sets to service customers, and use of multiple contact points within the team (Bendapudi and Leone 2002). In the offshore software services context, the service providers tend to negotiate with customers regarding acceptable replacements for employees who are leaving the project teams. The negotiation process leads to building of trust in the service provider and confidence that continuity and stability will be maintained in the ongoing project. In general, customers are less dissatisfied if they consider the replacement to be acceptable and have input into the replacement process (Bendapudi and Leone 2002).

While the arguments above establish the importance of team stability in the offshore software services context, two studies have specifically examined the link between employee turnover and customer behavior. In a study of turnover at a fast food chain, 20% of the outlets with the lowest employee turnover rates brought in double the sales and 55% higher profits than the 20% of stores with the highest turnover rates (Heskett et al. 1994). Likewise, in a study of convenience stores, a lowering of the employee turnover rates when levels of employee turnover were relatively low yielded significant improvements in customer satisfaction (Estelami and Hurley 2003). Building on these arguments:

H8: Team stability positively influences customer satisfaction.

In the context of offshore outsourcing, cultural differences between the service providers and customers can lead to communication problems (Kobayashi-Hillary 2005; Wright 2005). To facilitate effective communication, engineers in offshore software development firms are put though a training phase that enables them to understand the prevailing technology in use, the social etiquettes followed within both their own organization and customers organizations, and the native culture of the customer (Abdel-

Hamed 1989). The cross-cultural component of such training aims to improve communication skills by creating appropriate perceptions about the customer's culture (Black and Mendenhall 1990). However, such skills are seldom developed well during a short training course or in a formal learning environment. Learning about and becoming comfortable with the customer culture and developing the communication proficiency that fits in well with that culture takes time and experience (Torbiron 1982).

Specifically in the offshore context, new engineers who join the team take time to learn intricacies of communicating with customers. Communication skills in this context do not just include the ability to understand and operate in a different culture. In parallel, what is also called for is the ability to communicate in a mutually understandable technical language that reflects knowledge about the software application domain and the way that domain maps into the software design (Curtis et al. 1988). Consequently, effective communication is disrupted when experienced engineers leave and new engineers join the project team. Building on these arguments:

H9: Team stability positively influences communication effectiveness.

The stability of the project team can affect the quality of project management as well. When teams are stable, the software service provider can adhere to planned estimates and execute work without disruption, and better manage the changes in requirements and priorities – this enables superior project management (Abdel-Hamed 1989). When turnover is high, a key reason for the disruption of work planning is that there is limited information within the team on the new entrants' capabilities (Höffler and Sliwka 2002) – this leads to non-optimal work allocation and incorrect performance expectations. Second, the demarcation and management of project priorities is affected because team members, especially senior team members and managers, need to provide constant attention and guidance to the new entrants to help them quickly move up the learning curve (Chapman 1998). Further, the knowledge resident in the departing team members has to be documented and absorbed by others within the team. This is often a time-consuming task, given that a large fraction of the useful knowledge in software development is of a tacit nature. Finally, turnover decreases the project team's ability to recognize and manage sources of risk. New team members are typically slower than experienced ones in identifying problems at an early stage of development, and are less capable of taking remedial action that will resolve the problem at that early stage. Consequently, projects with high turnover are more likely to not meet their goals, be late, and suffer cost overruns (Abdel-Hamed 1989). Building on these arguments:

H10: Team stability is positively associated with superior project management

Control Variable: Team Size

The size of the project team can influence some of the constructs discussed above. We measure size as the total person months of effort expended on the project. While the number of employees associated with the project provides an alternative measure of size (e.g., Kraut and Streeter 1995), the metric we employ accounts for additions and attrition to the team over time, as well as the total effort input by employees into the project. Our field research showed that for very small projects that involve two or three employees, the tasks related to communication and project management were typically not well-defined. Instead, clients tended to dictate project needs on a task oriented basis. Employees in very small projects tended to focus more on day-to-day tasks and deadlines, making project management and communication harder. As the project size increases, the project lends itself to the imposition of greater structure and is better defined while still being manageable. This can help with improving communications, project tracking, and task estimation. However, as projects grow much larger, both communication and project management can become more difficult. For example, an empirical study of large projects revealed that employees found it easier to manage tasks when there were a smaller number of people involved (Curtis et al. 1988). With smaller project teams, a sub-group of team members can direct its work and closely track and manage project implementation (Kraut and Streeter 1995). With an increase in team size, the need for coordination and communication both within and outside the team can increase sharply, thereby decreasing the quality of project management and communications (Brooks 1995). As one programmer noted (Curtis et al. 1988, p. 1279): "In the beginning it was easy to keep track of what was going on. It was only after reaching the critical mass…that things began falling into cracks and we were losing track." Building on these arguments:

H11: Project size has an inverted-U shaped influence on project management.

H12: Project size has an inverted-U shaped influence on communication effectiveness.

2.3. Research Design

Research Setting and Data

Our dataset comes from a large, export-oriented, India-based software services company. We began the research with on-site field work conducted over two months at the company's operating sites in India. This immersion, which included interviews with company managers, helped us obtain an insider's view of the operation of outsourced offshore software projects.

Our unit of analysis is an offshore project team that provides software development, maintenance, and testing services to U.S.-based customers. Our data are sourced from projects that were executed from multiple sites of the software services provider. Our sample consists of 677 usable surveys filled in by U.S.-based managers employed by the customer who coordinated the offshore projects. These surveys evaluated services rendered by the offshore team over a 6-month span. The surveys captured perceptual ratings of the constructs discussed in §2 using 5-point Likert scales with 1 indicating strong disagreement, and 5 indicating strong agreement (see Appendix for survey items). As feedback was submitted for over 95% of the projects undertaken, non-response bias is not an issue.

The projects in our dataset can be classified in three ways. First, based on the nature of the task, projects could be classified into "Maintenance and Development" (M&D) or "Testing" projects. M&D projects involve fixing bugs, enhancing existing software features, and adding new features. Testing projects involve verification testing (automated testing procedure to ensure that bugs fixed in an existing software release did not disrupt other parts of the code) and test automation services (producing software test suites for new modules on which automated verification testing can be performed). Second, based on the type of software involved, projects could be classified as "System software" or "Application software" projects. The former runs close to and interacts with the hardware, while the latter runs on top of the system software. The skill sets involved in designing and managing systems and application software tend to be distinct. Finally, projects can be classified on the basis of duration. Projects with a functional duration of less than 2 years were classified as low age and those with more than 2 years were classified as high age projects. This classification was based on the field research and insights from the practitioner literature (Kaka and Sinha 2005). The maximum age of the project was close to 8 years since inception. The minimum age of the project in the dataset was 2 months.

Scale Validity and Reliability

Our final item list had 13 items, excluding an overall customer satisfaction measure (see Table 1 for items). We first performed an exploratory factor analysis (EFA) using varimax rotation on all survey items other than the overall satisfaction measure. The 12 items yielded five factors – these are labeled as Communication Ability, Communication Intensity, Project Management, Project Performance and Team Stability (see Tables 1 and 2). The five items together accounted for 74.2% of the total variance in the data. Subsequent EFA conducted separately on the M&D project data (381 surveys) and Testing project data (296 surveys) revealed consistent underlying factor structures.

A confirmatory factor analysis (CFA) was then performed on the factors that emerged from the EFA, applying a Robust Maximum Likelihood (RML) estimator to correct for multivariate non-normality (using LISREL 8.72). All the data, including the combined dataset and subsets of the data described fit the model well (see Table 3), and the loadings of the indicators on each of the constructs were significant for each dataset. Further, as detailed in Table 1, each construct in the combined dataset had a Cronbach’s alpha that was well over the 0.70 criterion (Nunnally 1978).

Finally, following Venkatraman (1989) and Sethi and King (1994), we tested the constructs for discriminant validity by comparing an unconstrained model with each pair of constructs grouped together and a model with the covariance between the two constructs constrained to unity. A significant difference in Chi-Square between the models indicated that the constructs were distinct (see Table 4). Overall, these findings suggest that the measurement model performs well.

Construct	Items	Alpha
Project Management	a) Work planning and estimation b) Managing changes in project schedules/priorities c) Risk identification and management	0.82
Project performance	d) Overall Quality of delivery e) Overall Timeliness of delivery f) Managing interim Goals	0.78
Communication Intensity	g) Quality of status reports h) Timeliness of reports	0.809
Communication Quality	i) Oral communication ability j) Written communication ability	0.785
Team Stability	k) Satisfactory duration of stay of engineers in team l) Satisfactory management of transitions within team	0.805

Table 1 # Scale and Reliability measures using combined data

	Project Management	Project Performance	Communication Intensity	Communication Ability	Team Stability
a)	**0.7728**	0.3372	0.2107	0.1143	0.0886
b)	**0.7172**	0.1596	0.3124	0.2996	0.1729
c)	**0.7012**	0.2413	0.1419	0.311	0.1637
d)	0.3484	**0.6687**	0.072	0.2368	0.1471
e)	0.5364	**0.6532**	0.1442	0.0665	0.1523
f)	0.1443	**0.8025**	0.24	0.1495	0.1915
g)	0.2815	0.1208	**0.832**	0.2063	0.1144
h)	0.1704	0.2179	**0.8339**	0.2485	0.0794
i)	0.2189	0.1338	0.2054	**0.8625**	0.0746
j)	0.2744	0.2341	0.3091	**0.7338**	0.143
k)	0.1214	0.1414	0.0707	0.039	**0.884**
l)	0.1434	0.1647	0.101	0.1403	**0.8507**

Table 2 # Factor loadings for Combined Data

	Sample Size	CHI - Square	DF	RMSEA	NFI	IFI	RFI	GFI
Combined Data	677	93.74	44	0.049	0.985	0.99	0.978	0.971
Maintenance and Development	381	71.61	44	0.04	0.978	0.988	0.967	0.962
Testing	296	64.86	44	0.04	0.979	0.989	0.969	0.952
Systems Software	450	72.109	44	0.0377	0.984	0.991	0.977	0.966
Application Software	227	71.55	44	0.052	0.965	0.981	0.948	0.938

Table 3 # Results of CFA analysis for the classifications

	Constrained Model	Unconstrained Model	DF Difference	CHI-Square Difference	P-Value
Project Management with					
Project Performance	194.00	38.89	1	155.11	<.001
Team Stability	174.39	1.59	1	172.80	<.001
Communication Intensity	172.30	3.20	1	169.10	<.001
Communication Ability	174.07	5.21	1	168.86	<.001
Project Performance with					
Team Stability	205.68	4.65	1	201.03	<.001
Communication Intensity	207.16	5.75	1	201.41	<.001
Communication Ability	302.54	6.14	1	296.40	<.001
Team Stability with					
Communication Intensity	357.52	0.00	1	357.52	<.001
Communication Ability	368.86	0.05	1	368.81	<.001
Communication Intensity with					
Communication Ability	264.80	1.24	1	263.56	<.001

Table 4 # Difference in chi square from a constrained model with covariance 1 and free model

2.4. Analysis and Results

Analysis Methodology

We used a structural equations model to validate our hypotheses. We analyzed the covariance matrix of the combined data (N = 677) using LISREL 8.72. While most of the variables displayed either insignificant or only moderate kurtosis, an examination of multivariate kurtosis indicated a high Mardia's coefficient of 90.03. We used the Satorra-Bentler scaled Chi-Square and a Robust Maximum Likelihood (RML) estimator to correct for the downward bias in standard error – this is the most reliable correction for non-normality (Boomsma and Hoogland 2001; West et al. 1995). An alternative correction approach is to use an Asymptotic Distribution Free (ADF) estimator. A simulation study conducted by Olsson et al. (2000) compared maximum likelihood, Generalized least squares

(GLS) and ADF estimators. They concluded that maximum likelihood is most robust to specification errors and non-normality. ADF estimators yield unbiased estimates and approached ML estimators in performance only with sample sizes of over 1000 observations. Therefore, we use the RML estimator.

Combined Data Results

Descriptive statistics for the combined data set (N=677) are provided in Table 5. All item-to-construct R-squares are greater than 50% (see Table 5). Table 6 details the multiple R-Squares of the endogenous constructs – note that 71.52% of the variance in customer satisfaction is explained by the model. Estimation results for the combined data and other sub-classifications of the data are detailed in Table 7. The estimated model has an RMSEA of less than 0.05 across the data classifications, indicating good model fit (MacCallum et al. 1996). All the comparative fit indices are greater than 0.9 across the classifications, again indicating good fit (compared to independence model). Indirect and the total effects between constructs, and the significance of the corresponding paths are reported in Tables 8 and 9[4].

Focusing first on the results for the combined data set, we find that project performance had a significant positive impact on customer satisfaction (i.e., H1 is supported). Project management had a significant positive impact on project performance and customer satisfaction (i.e., H2 and H3 are supported). Project management had a direct (positive) impact on customer satisfaction, and also an indirect impact on customer satisfaction through project performance. Stated differently, developing project management capabilities appears to improve project performance, and hence customer satisfaction. Increased customer satisfaction has been associated in past literature with increased customer retention and profits. This supports the notion that project management capability drives firm profits in offshore software development (Ethiraj et al. 2005).

Next, we find that communication effectiveness has a significant, positive impact on project management (i.e., H4 is supported), and this effect holds across all data classifications. Further, project management fully mediates the impact of communication effectiveness on project performance across all data classifications (i.e., H5 is supported). Other empirical findings in the literature have also revealed the lack of a direct effect of communication on project performance (Deephouse et al. 1996). However, a similar mediated effect is found by Nidumolu (1995), who finds that the effect of horizontal coordination (oral communication, written communication, scheduled and unscheduled group meetings) on project performance is fully mediated by residual risk. Further, we did not find a significant direct impact of communication effectiveness on customer satisfaction (i.e., H6 is not supported) – however, its indirect effect through project management and project performance is positive and significant across all the data classifications (see Table 8).

---

4 Based on the modification indices generated by the estimation, we allow for free inter-item correlation between work planning and estimation, and the timeliness of delivery across datasets.

		(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)	(m)	(n)
Plan	(a)	1.000													
Prioritize	(b)	0.623	1.000												
Manage Risk	(c)	0.579	0.619	1.000											
Delivery quality	(d)	0.514	0.457	0.513	1.000										
Delivery Timeliness	(e)	0.658	0.558	0.514	0.569	1.000									
Interim goals	(f)	0.462	0.460	0.470	0.476	0.591	1.000								
Status Rep Quality	(g)	0.447	0.527	0.442	0.344	0.370	0.389	1.000							
Status rep timeliness	(h)	0.428	0.492	0.406	0.374	0.388	0.390	0.686	1.000						
Oral ability	(i)	0.399	0.508	0.452	0.363	0.347	0.340	0.456	0.454	1.000					
Written ability	(j)	0.484	0.554	0.516	0.434	0.439	0.419	0.509	0.542	0.652	1.000				
Duration of Stay	(k)	0.245	0.295	0.282	0.297	0.318	0.309	0.215	0.203	0.184	0.242	1.000			
Transition	(l)	0.305	0.332	0.326	0.319	0.344	0.356	0.263	0.249	0.245	0.331	0.620	1.000		
Overall Satisfaction	(m)	0.599	0.625	0.577	0.632	0.641	0.581	0.449	0.492	0.419	0.513	0.388	0.462	1.000	
Project Size (log)	(n)	0.056	0.094	0.067	0.015	0.045	0.037	0.026	0.107	0.045	0.018	-0.077	-0.006	0.071	1.000
Mean		4.423	4.586	4.323	4.481	4.536	4.292	4.573	4.661	4.419	4.599	4.179	4.226	4.543	3.236
Stdev		0.640	0.587	0.682	0.606	0.622	0.705	0.596	0.514	0.617	0.537	0.827	0.804	0.577	0.844
R-Square		0.577	0.664	0.578	0.523	0.616	0.502	0.687	0.686	0.561	0.757	0.547	0.703		

Table 5 # Item statistics and correlation matrix

Communication Effectiveness	0.2215
Communication Intensity	0.6798
Communication Ability	0.7841
Overall Satisfaction	0.7152
Project Performance	0.7578
Project Management	0.7818

Table 6 # R-Square for the endogenous constructs

	Combined	M&D	Testing	Application	System	Low Duration	High Duration
Sample Size	**677**	**381**	**296**	**227**	**450**	**242**	**435**
H1	0.667***	0.579***	0.764***	0.547***	0.714***	0.631***	0.771**
H2	0.741***	0.839***	0.644***	0.649***	0.783***	0.668***	0.797***
H3	0.285**	0.626**	-0.035	0.307	0.301	0.316	0.193
H4	1.217***	1.124***	1.284***	1.083***	1.292***	1.377***	1.146***
H5	0.903***	0.944***	0.827***	0.704***	1.013***	0.921***	0.915***
H6	-0.063	-0.440	0.364**	0.120	-0.220	-0.101	-0.017
H7	0.131**	0.078*	0.194**	0.234***	0.082*	0.224**	0.071*
H8	0.097**	0.097**	0.073	0.082	0.141***	0.063	0.108**
H9	0.233***	0.184***	0.288***	0.226***	0.235***	0.245***	0.209***
H10	0.095**	0.110**	0.080	0.145**	0.064	0.019	0.149***
H11	-0.292**	-0.099	-0.377**	-0.179	-0.303**	-0.192	-0.213
H11a†	0.050**	0.019	0.061**	0.029	0.053**	0.024	0.044**
H12	0.265**	0.409**	0.179	0.056	0.331**	0.084	0.313**
H12a††	-0.036**	-0.066**	-0.020	-0.007	-0.044**	-0.005	-0.046**
From MI	0.055***	0.053**	0.057**	0.029**	0.066***	0.092	0.035***
Chi-Square	98.440	80.839	107.694	96.307	80.982	101.682	98.399
DF	74	74	74	74	74	74	74
RMSEA	0.022	0.015	0.039	0.0365	0.014	0.039	0.027
NFI	0.991	0.985	0.982	0.974	0.990	0.977	0.986
RFI	0.988	0.979	0.975	0.963	0.985	0.968	0.980
CFI	0.997	0.998	0.994	0.993	0.999	0.993	0.996
GFI	0.977	0.967	0.944	0.938	0.972	0.937	0.967
NNFI	0.997	0.998	0.992	0.991	0.998	0.991	0.995

Table 7 # Fit indices and Direct effect estimates for the individual categories of dataset

†non-linear coefficient for H11 to test the nonlinearity hypotheses
†† *nonlinear coefficient for H12 to test the nonlinearity hypotheses*

** Significant at <0.1*
***Significant at <0.05*
****Significant at <0.001*

Indirect effects between the latent variables							
	Combined	**M&D**	**Testing**	**Low Age**	**High Age**	**System**	**Application**
Communication Effectiveness → Project Performance	0.903***	0.944***	0.827***	0.920***	0.914***	1.013***	0.703***
Communication Effectiveness → Customer Satisfaction	0.950***	1.252***	0.587***	1.016***	0.926***	1.113***	0.718***
Team Stability → Project Management	0.284***	0.207***	0.370***	0.337***	0.240***	0.304***	0.245***
Team Stability → Project Performance	0.281***	0.267***	0.290***	0.239***	0.310***	0.289***	0.254***
Team Stability → Customer Satisfaction	0.369***	0.318***	0.460***	0.380***	0.366***	0.325***	0.415***
Project Management → Customer Satisfaction	0.495***	0.486***	0.493***	0.421***	0.615**	0.560***	0.355**
Project Size → Project Performance	0.022	0.303**	-0.094	-0.050	0.116	0.097	-0.076
Project Size square → Project Performance	0.004	-0.046*	0.022	0.011	-0.007	-0.003	0.013
Project Size → Project Management	0.322**	0.460**	0.231*	0.116	0.359**	0.428**	0.061
Project Size square → Project Management	-0.044**	-0.074**	-0.026	-0.007	-0.052**	-0.057**	-0.007

Table 8 # Indirect effects between the latent variables

** Significant at <0.1*
***Significant at <0.05*
***Significant at <0.001

Total effects between the latent variables							
	Combined	**M&D**	**Testing**	**Low Age**	**High Age**	**System**	**Application**
Communication Effectiveness → Project Performance	0.903***	0.944***	0.827***	0.920***	0.914***	1.013***	0.703***
Communication Effectiveness → Customer Satisfaction	0.886***	0.811***	0.951***	0.915***	0.909***	0.893***	0.838***
Team Stability → Project Management	0.379***	0.318***	0.450***	0.357***	0.389***	0.369***	0.391***
Team Stability → Project Performance	0.413***	0.345***	0.485***	0.463***	0.382***	0.371***	0.488***
Team Stability → Customer Satisfaction	0.466***	0.415***	0.533***	0.444***	0.474***	0.466***	0.498***
Project Management → Customer Satisfaction	0.781***	1.114***	0.457***	0.738***	0.808***	0.861***	0.663***
Project Size → Project Performance	0.022	0.303**	-0.094	-0.050	0.116	0.097	-0.076
Project Size square → Project Performance	0.004	-0.046*	0.022	0.011	-0.007	-0.003	0.013
Project Size → Project Management	0.030	0.361**	-0.146	-0.075	0.145	0.124	-0.118
Project Size square → Project Management	0.006	-0.055*	0.035	0.017	-0.008	-0.004	0.021
Project Size → Customer Satisfaction	0.006	0.222	-0.001	-0.064	0.112	0.034	-0.071
Project Size square → Customer Satisfaction	0.007	-0.032	0.008	0.013	-0.006	0.006	0.013
Project Size → Communication Effectiveness	0.265**	0.409**	0.179	0.084	0.313**	0.331**	0.056
Project Size square → Communication Effectiveness	-0.037**	-0.066**	-0.020	-0.005	-0.046**	-0.044**	-0.007

Table 9 # Total effects between the latent variables

** Significant at <0.1*
***Significant at <0.05*
***Significant at <0.001

As hypothesized, the effects of team stability on project performance (H7), customer satisfaction (H8), communication effectiveness (H9) and project management (H10) were positive and significant.

Counter to our original hypotheses of an inverted-U shaped effect of project size on project management (H11), we find a U shaped effect.[5] A possible, ex-post explanation that must be considered exploratory in nature is as follows. Our field research revealed that with very small project teams in the offshore context, project management responsibility was shared across multiple projects by a single offshore project manager. In such cases, the offshore project manager tends to develop deep skills in managing multiple, small projects. However, as the project size increases, the ability of this single offshore leader to focus sufficient attention and resources on any given project decreases—this leads to a deterioration in how tightly the project is managed. However, as the project grows in size, a dedicated team manager is allocated to that project team. This may lead to a more systematic and disciplined approach to work planning, resource allocation, and risk management, thereby enabling a tighter overall management of the project. This reasoning would explain the U-shaped effect of project size on project management. The indirect effect of project size on project management through communication effectiveness is significant and in the expected direction. Finally, we find support for the hypotheses that project size has an inverted U shaped effect on perceived communication effectiveness (H12).

Finally, Table 10 shows a comparison of the coefficient estimates of our work with comparative software studies. It also displays the context in which the referred research was

[5] We used a log transform of the project size. First, this allows the size variable to be similarly scaled as the other indicators. Second, this normalizes the size variable and reduces the level of multivariate non-normality in the data.

carried out. While one must acknowledge that it is not an apple to apple to comparison, it does carry some value as we may infer from the following discussion. First, as seen in Table 10, one straight forward inference that we can draw is that team stability has not been considered in past literature. Next, our results match the results found in other outsourcing studies where the investigation is in non-offshore specific contexts. Of specific interest in Table 10 is the lack of strong correlation between communication and residual risk as reported by Nidumolu (1995) as opposed to a strong correlation as found in our data. As Nidumolu's research context was not offshore, this may point to better project management abilities required in the offshore software development context. In similar vein, Nidumolu and Subramani (2003) did not find any effect of the standardization of processes on the performance of the project while we find a direct impact of project management (a proxy for standardization) on project performance, again reinforcing the need for effective project management skills in offshore environment. This is also consistent with findings of Ethiraj et al. (2005) who find that project management is a significant capability that offshore software developers need to manage and develop.

Other interesting results show that the quality of deliverables was not significantly associated with client satisfaction in Kraut and Streeter (1995) while we find a significant association. Further, we found a nonlinear effect of project size on communication and on project management as opposed to a linear effect in other related literatures such as Kraut and Streeter (1995). Finally, our investigation of the non-linear effect of project size on project management has not been investigated in the literature and the results show a counterintuitive effect that is consistent with some of the insights that we obtained from the managers on the field.

	Customer Satisfaction	Project Management	Project Performance
Customer Satisfaction	**N/A**	This link has not been directly studied	(-) 0.11 **(0.850***)** (Client satisfaction and Software quality - Kraut and Streeter 1995) - **non offshore context SOFTWARE**
Project Management		**N/A**	0.702* **(0.658***)** (Elapsed time in project to Technical Processes, Gopal et al. 2002) - **offshore context SOFTWARE** (-) .20 **(0.741***)** (Standardization of Methods to Development Process performance, Nidumolu and Subramani 2003) - **non offshore context SOFTWARE** 0.791** **(0.658***)** (Meeting Targets and Planning - Deephouse et al. 1996) **- Non offshore context SOFTWARE** 0.228* **(0.514***)** (Overall Quality and Planning - Deephouse et al. 1996) - **Non offshore context SOFTWARE** (-) 0.54*** **(0.741***)**† (Residual Risk and Project performance -- Nidumolu 1995) - N**on offshore context SOFTWARE**
Project Performance			**N/A**
Communication			
Team Stability			
Team Size			

Table 10 # Comparison of estimates from common model with similar work in software or services domain[6]

[6] Standardized and unstandardized coefficients are compared where appropriate

	Communication	**Team Stability**	**Project Size**
Customer Satisfaction	0.610*** **(0.886***)** (Coordination Success and Client Satisfaction - Kraut and Streeter 1995) **non offshore context SOFTWARE**	Not studied in earlier literature in this context	(-) 0.05 **(0.071)** (Project Size and Client Satisfaction – Kraut and Streeter 1995) **non offshore context SOFTWARE**
Project Management	(-) 0.20 **(1.217***)** (Horizontal coordination and Residual project risk - Nidumolu 1995) **non offshore context SOFTWARE** 0.35** **(1.217***)** (Extra-Project Communication and Customer Uncertainty - Lievens and Moenaert 2000) **FINANCIAL SERVICES DESIGN**	Not studied in earlier literature in this context	0.32** **(0.1461**)** (Team size to formal impersonal procedures - Kraut and Streeter 1995) **non offshore SOFTWARE**
Project Performance	0.38** **(0.903***)** (Horizontal coordination and Project Performance - Nidumolu 1995) **non offshore context SOFTWARE** 0.510*** **(0.344***)** (Reporting Quality and Project Performance - Thompson et al. 2007) **non offshore context SOFTWARE** -0.050 **(0.576***)** (Coordination success and software quality - Kraut and Streeter 1995)	Not studied in earlier literature in this context	(-) 0.052 **(0.045)** (Team size to project schedule slippage) (Ethiraj et al. 2005) **Offshore context SOFTWARE**
Communication	**N/A**	Not studied in earlier literature in this context	(-0.06) **(0.364**)** (Team size to coordination success) (Ethiraj et al. 2005) **Offshore context SOFTWARE**
Team Stability		**N/A**	
Team Size			**N/A**

Table 10a # Comparison of estimates from common model with similar work in software or services domain†

† * Significant at < 0.1, **Significant at < 0.05, ***Significant at < 0.001
†† Bold numbers in the parenthesis in each of the cells corresponds to the estimates obtained in our study

Results of the Contrast Models

We now compare the empirical findings across the data classifications.

Maintenance and Development (M&D) Versus Testing Projects

While the model fits both M&D and testing project data well (see Table 7), there is a significant direct impact of project management on customer satisfaction (H3) in M&D projects, but not in testing projects. A possible explanation is based on the difference in task uncertainty – the level of unpredictability of the task the engineer is engaged in – across these two project types. Task uncertainty is higher in M&D projects because it involves understanding and fixing defects, and developing or enhancing software functionality (Brooks 1995; Pressman 2005). These tasks are typically not well structured and are subject to both evolving customer requirements and frequent customer involvement. Therefore, prioritization, work planning, and resource allocation need to be handled particularly well. Accordingly, for M&D projects, project management is of separate importance in itself, and directly impacts customer satisfaction.

On the other hand, task uncertainty is low in testing projects that involve verification testing and test automation services. Verification testing activities follow well-defined test routines specified by the customer and involve clear reporting requirements. Test automation work involves the development and execution of standard test scripts to automate the verification testing process. Because these tasks are highly structured and involve predictable time and resource requirements, ongoing prioritization, work planning, and resource allocation may not be so crucial. Further, the actual involvement of the customer with the team during the project is quite limited since most of the testing process occurs with little customer interaction with a report provided at the end of testing. As a result one would expect that project management may not directly impact customer satisfaction in testing projects—instead, its effect is mediated by project performance.

Second, we find that communication effectiveness has a significant direct impact on customer satisfaction in testing projects, but not in M&D projects. However, for both project types, communication effectiveness has a significant ($p<.01$) indirect impact on customer satisfaction (H6). Arguably, the regular reporting of progress and testing results to customers is of independent importance in testing projects because of the highly structured and time-bound nature of the tasks involved.

Third, the direct effect of team stability on project management (H10) is significant ($p<0.01$) for M&D projects but not for testing projects. For M&D projects, learning is slow because the engineer has to gain familiarity with the code base and the software application domain to resolve defects and develop new features compatible with existing code (Banker and Slaughter 1997; Littman et al. 1987). In contrast, for testing projects, considerable work content is automated and the learning is faster. Further, once new engineers have had a limited initiation to the project, they can easily run predesigned testing scripts and interpret the results.

Application Software vs. System Software Projects

There is a significant direct effect of team stability on customer satisfaction (H8) in system software projects but not in application software projects. Our field research revealed that it was more difficult to find and replace engineers who were skilled in systems software, compared to application software. The relative scarcity and strong demand for talent in the system software domain has also documented by other researchers and practitioners (Bland 2006; Clair and Linden 2005). Our finding is consistent with this perspective.

The other major difference between these two classifications is that there is a direct, significant effect of team stability on project management (H10) in application software projects, but not in system software projects. Our discussions during the field studies and post-analysis interviews revealed that teams in system software projects were focused on comparatively small enhancements to large, complex pieces of code. In contrast, the teams engaged in application projects were involved in developing larger pieces of code. This called for greater coordination between engineers working on separate code components and across different phases of the development cycle. Further, teams working on application software projects assumed a large proportion of the total project responsibility and were typically responsible for the management of all development cycle activities. Consequently, these projects tended to last longer. Under these conditions, stable teams enabled superior long-term management of the projects.

High Age Vs Low Age Projects

Team stability has a significant direct impact on customer satisfaction (Hypotheses H8) in older projects, but not in newer projects. Intuitively, in longer duration projects the offshore teams acquire greater knowledge about the product and have a successful history of managing the development process. Our field research revealed that in long duration projects the offshore engineers drew on this experience history to contribute more substantially to product enhancements, often exceeding the expectations of customers. Therefore, customers are particularly sensitive to the turnover of experienced hands in long duration projects because the loss of tacit knowledge in the development team and the corresponding productivity drop may be substantial. In addition, customers tend to build a personal rapport with engineers they have known for a while, and are averse to building new relationships with incoming employees.

The sensitivity of customers to turnover in long duration projects may be exacerbated by the fact that most turnover does occur in such projects. Turnover in the Indian software industry is generally high – on average, only about 25% of hired engineers continue with a company for more than 5 years (Sudhakar 2002). An analysis of turnover from about 150 projects within our dataset revealed that about 75% of the total turnover occurred in high age projects and involved engineers with an average experience of 1.5 years.

We also find that team stability has a significant direct influence on project management (H10) in high age projects, but not in low age projects. In newer projects that are yet to stabilize, the teams go through a learning curve that enables them to understand the code and the interactions between various software components (Fjeldstad and Hamlen 1983;

Littman et al. 1987). Such an understanding, which comes with time, can help in planning, estimation and risk management activities (Rajlich 1999). Once such experience is built into the team over a period of time, team stability then plays an important role in securing that knowledge, so that it can be leveraged towards better managing the project.

2.5. Conclusion

Our findings have multiple implications for the management of outsourced offshore software projects. First, the study suggests that managing the offshore projects entail a life cycle effect where different capabilities are more important for different situations. For offshore managers, we find that team stability is an important component affecting project performance, project management, communication effectiveness and customer satisfaction. In general, managers must work hard to maintain the stability of the project teams. However, the overall duration of the offshore project moderates some of the influence of team stability. For new projects, team stability influences project performance, but does not directly influence customer satisfaction. In contrast, for older projects, team stability directly impacts customer satisfaction. Intuitively, when projects are new the customer is focused mainly on project delivery. However, as the team gains experience and project management capabilities over time, customers are loathe to lose valuable team members, and turnover directly affects customer satisfaction. While offshore managers must strive to maintain stable teams throughout, such stability is of even greater importance in established projects.

This finding also has implications for a practice that several software services firms' use called 'shadowing.' Shadow engineers work with project groups to learn the work, and prepare themselves to be inducted into the project. While the practice of shadowing can help stabilize the project by allowing a new engineer to begin taking over work from an existing engineer who is anticipated to move out of the project and can help cut training lead times, it involves a short-term duplication of resources within the project team. In practice, shadow engineers are employed in new projects – often at project inception – as well as in long duration projects. Our results suggest that managers may best employ 'shadow engineers' not in the newer projects, but in projects that have been established (Greater than 2 years). This gestation period allows for knowledge structures within the project to be stabilized, and this knowledge can then be more readily transferred to the shadow engineers.

Project management capabilities play an important role. They improve project performance and also impact customer satisfaction. Project management capabilities also mediate the impact of communication effectiveness on project performance. For managers of offshore software firms, this implies that a sole focus on hard technical capabilities related to software design and programming is insufficient; adequate attention must be paid to developing capabilities that enable effective planning and management of resources across time, and also managing communications with customers. The achievement of benchmark assessment ratings that are designed to build project management capabilities, including the well-known Capability Maturity Model from the Software Engineering Institute (SEI-CMM), can help in this context. The role of project management capabilities is more important when project-related uncertainty is high. Accordingly, managers must seek to build such capabilities particularly for M&D projects. Our results suggest that such capabilities are particularly effective in driving customer satisfaction in the uncertain environment associated with M&D projects (as opposed to testing projects).

Communication effectiveness positively affects perceptions of project performance and project management. While communication effectiveness does not directly affect customer satisfaction in most cases, its indirect effect through project performance and project management is significant. Our results suggest that managers must pay attention to multiple components of communication – including communication intensity and quality – towards delivering a satisfactory customer experience. Our interviews of managers at offshore software service providers suggested that some of these providers managed communications with customers better than others. In firms that performed strongly on this dimension, the offshore managers usually drew up a formal communication plan for new projects and these plans were reviewed periodically to reflect the needs of the prevailing project situation. These plans specified the reports that were to be exchanged, the lines of communication that were to be maintained, communication formats (email/oral communication), issue escalation hierarchies, the preferred initiators of communication for various tasks, preferred times for conference calls, and also the communication plan review periods.

We find evidence of non-linear, inverted-U shaped effects of team size on communication effectiveness. This suggests that effective communication with customers can be an issue in both very small and very large projects – though for different reasons, as explained earlier. Correspondingly, managers must play special attention to implementing procedures that support the communication process in such projects.

While we have discussed some of the key managerial implications of our findings from the perspective of the offshore managers, much of the discussion applies, with due adjustments, to customers in selecting outsourcing partners and managing outsourcing projects. For example, our findings suggest for tasks characterized by high uncertainty, the customer must try and choose a service provider with strong project management capabilities. Likewise, the customer must set expectations of performance depending on the contextual variables associated with the project. For example, since learning takes time, we find that team stability is less crucial in the early stages of a project but is more important for projects that have been in the works for a while. Instead of being concerned about the lack of team stability during the very early stages of a new project, the customer may benefit from focusing on other commitments related to delivery specifications, quality, and timeliness.

CHAPTER 3

3. Individual Learning and Productivity in a Software Maintenance Environment: An Empirical Analysis

3.1. Introduction

Consistent with Adam Smith's championship of the division of labor, firms have long viewed task specialization and experience on the job as a stepping stone to enhanced employee learning and productivity. To understand how such experience-based learning impacts productivity, researchers have studied multiple perspectives of the learning phenomenon, including the estimation of learning rates (e.g., Dutton and Thomas 1984) , the temporal patterns of learning (e.g., Lapré and Tsikriktsis 2006), and the accumulation and transferability of experience (Garg and Milliman 1961). Researchers have examined the relationship between experience and productivity across a range of work contexts – these include the airline (Asher 1956), apparel (Baloff 1971), semiconductor (Hatch and Mowery 1998), manufacturing (Lapré and Van Wassenhove 2001), and service industries (Reagans et al. 2005). In general, there is strong evidence regarding the positive implications of specialization and experience for learning and productivity.

At the same time, researchers and managers have become sensitive to the fact that greater specialization can sometimes hurt, rather than help. For example, firms that are highly specialized and inward looking can be overly focused on strengthening and exploiting existing competencies. These firms can suffer from "learning myopia" and may overlook the future (Levinthal and March 1993). Consequently, they can stumble into "competency traps" where their own competencies ultimately limit their external, long-term vision. Organizations can steer clear of such traps and build new competencies by developing greater flexibility and absorptive capacity – the ability to store, retrieve, and transfer knowledge (Abernathy and Wayne 1974; Cohen and Levinthal 1990).

While these arguments have been explored mainly at an organizational (or macro) level, some interesting, parallel arguments exist at the micro level of individual employees and teams (Cohen 1991).[7] Specifically, how does the pattern of past experience enhance individual learning and productivity? While Adam Smith championed the notion of specialization, is there something such as employee overspecialization? How do the specialization of experience (i.e., substantial work at a specific task) and the variety of experience (i.e., work that is dispersed across a variety of tasks) affect learning and productivity? How do specialization and variety of experience interact with each other in driving learning and productivity? This work contributes to the literature by examining these issues.

The empirical investigation of these questions at the level of the individual employee has been limited (Argote et al. 2003). In the manufacturing context, several authors have found that cross training enhances productivity – see Hopp and Oyen (2004) for a review. Likewise, the implications of product variety for organizational performance have been extensively studied – see Ramdas (2003) for a review. However, relatively little is known about the overall impact of task variety on individual learning and productivity (Boudreau et al. 2003). The work of Schilling et al. (2003) constitutes a notable, recent exception. Schilling et al. designed an experimental setting where study participants were classified

[7] See Schilling et al. (2003) for a brief review of the literatures related to organizational, group, and individual learning.

under three conditions: specialization (playing multiple rounds of the strategic board game *Go*), related variation or variety (mixing up playing *Go* with playing the similar board game *Reversi*) and unrelated variation (mixing up playing *Go* with a computer version of the unrelated card game *Cribbage*). Their findings indicate that: (a) cumulative experience (number of Go games played) improved points scored on *Go*; (b) related experience with *Reversi* enhanced scores on *Go*; (c) related variation has a stronger effect on performance on *Go* than either specialization or unrelated variation; and (d) specialization and unrelated variation similarly affected performance on *Go*. [8]

Our work builds on this and other work in the learning literature in the following ways. First, as noted by Schilling et al., real-life problems may be open ended and poorly defined, compared to an experiment that involves a well-defined strategic problem with immediate and accurate performance feedback. We use a panel data set that captures the performance of software engineers in a single firm on a range of debugging tasks to explore the validity of existing findings in the learning literature, and to deliver some new insights regarding learning and productivity.

Second, we examine how specialization and variety of experience interact with each other in driving learning and productivity. This allows us to explore how specialization and variety of experience independently and interactively drive learning and productivity, and when there is "too much variety." In this context, we adapt and apply the concept of

[8] Learning and productivity are two sides of the same coin. Learning results in increased storage of information in memory, and in the application of superior perceptive and cognitive processes that operate on that information. At this level, learning is invisible and immeasurable. However, to be meaningful, learning must ultimately manifest itself in superior task performance, whether that task involves taking a test or working on a machine. Consistent with the literature, we treat such superior task performance at an individual level as an evidence of learning (e.g., Huntley 2003; Schilling et al. 2003).

Herfindahl-Hirschman Index from the economics literature towards understanding the trade-off between specialization and exposure to variety.

Third, there is a need to examine how contextual variables in real-life settings impact learning and productivity (Schilling et al. 2003). One of the key variables that impacts learning rates in practice is the stability of the project team. Argote (1999, p. 54) notes: "Although there has been a long tradition of research on predictors of turnover, only recently has research been devoted to determining the consequences of turnover." While previous studies have typically not found a strong effect of turnover on learning outcomes, this may be because the study context may play an important moderating role (Argote et al. 1990; Argote 1999). In this study, we investigate how both the entry of new team members and the exit of existing team members impact learning and productivity.

Finally, much of the learning literature, and in particular the literature related to experience curve effects, has focused on manufacturing industries. The importance of the software services sector has steadily grown as firms and economies have become more knowledge-intensive, and as communication networks have grown in capacity and capability (Gopal et al. 2002). Apart from the theoretical contributions, our work yields a deeper substantial understanding of employee learning and productivity in the services sector.

Data for the study was sourced from a large export-oriented software services provider based in India. The firm employs over 10,000 engineers and provides development and maintenance services in application and system software domains. Software-related exports comprised more than 90% of the firm's revenue. The research effort was launched with on-site field work conducted over two months at the firm's operating sites in India. The firm and its customers operated in a highly competitive environment, and were faced with substantial challenges related to maintaining and supporting existing and newly introduced products. The firm's operations reflected the standard elements of competitive and dynamic business environments, viz., shortening product cycle times, increasing feature introduction velocity, and rising product proliferation (Krishnan et al. 1998; Swaminathan and Lee 2003; Swaminathan and Tayur 2003). The firm's customers progressively outsourced greater responsibilities to the firm in order to free up their own internal resources for more advanced projects – this is consistent with observations in other industry sectors (Nitsch and Swaminathan 2006). Whereas the firm was growing rapidly, it was also challenged to work efficiently with available resources in a tight market for skilled engineers.

In this environment, top management valued rapid on-the-job learning by each engineer. Managers also believed that useful learning derived mostly from experience on the job, and not from in-class training. Further, given the rapid rate of product modification and introduction, maintenance teams were required to continuously work on newer tasks and technologies. Therefore, exposure to task variety was considered a key component of learning. In parallel, managers also recognized than intensive specialization would lead to motivational problems. Exposure to a greater variety of tasks on the job has been associated with lower fatigue, reduced absenteeism, and correspondingly, with increased productivity (Cappelli and Rogovsky 1994). Managers we interviewed noted that some engineers were dissatisfied after being on a project for as little as a year. Engineers also shied away from substantial specialization in order to build a portfolio of skills that would enhance their worth in the external market and stature within the firm (Loch et al. 2000).

The firm also grappled with employee turnover, a key concern for the software industry worldwide (Amoribieta et al. 2001). The managers in the firm realized that effective and quick learning was indispensable towards maintaining a steady inventory of capabilities in the face of such churn.

The field research was accompanied by the collection of archival information, which was later compiled into a panel data set. These data captured details regarding the time of allocation of specific software defects to specific engineers across time, the time to resolution for the defects, defect characteristics, the software modules the defects occurred in, and engineer characteristics.

Our key findings are as follows. First, cumulative experience enhances learning and productivity. Second, there is an additional productivity gain for an individual that accrues from performing a greater variety of tasks. However, training individuals on such variety leads to a short term loss in productivity. This loss can be viewed an investment towards accruing long term productivity gains. Third, there is “too much variety” – we demonstrate that there is a tradeoff between exposure to a variety of modules and specializing on the modules that one is exposed to. Correspondingly, we find that variety plays an important role in increasing productivity only when the experience of the individual is highly specialized or concentrated. Fourth, we find that more experienced individuals can handle new tasks faster than those with lesser experience, i.e., experience gained in the software maintenance context can be transferred and applied to the performance of new tasks. Fifth, individuals in larger teams are more productive than individuals in smaller teams. This suggests that informal collaborative environments where collocated individuals possess diverse but related skills can enhance learning. Finally, in the context of team turnover, individuals in smaller teams bear greater productivity loss due to employee joining compared to individuals in larger teams. We also find that, surprisingly, entry of new employees joining may lead to a greater productivity loss than employee exit.

The remainder of the work is structured as follows. The hypotheses are developed in § 2. The data and measures are presented in § 3. The model and the estimation method is presented in § 4. Empirical findings are detailed in § 5 – some robustness checks are also described here. The research and managerial implications of our findings are described in § 6.

3.2. Hypotheses

We first discuss the hypotheses that relate individual experience to productivity, and then those that relate team changes (turnover) to productivity. Our dependent variable is the average length of time taken by an engineer to debug software defects allocated to him or her at a certain point in time.

Cumulative individual experience

There is a wide consensus among researchers and managers that more experience at a focused or specialized task yields higher productivity. This linkage, which was first prominently highlighted by Adam Smith (1776) was used to popularize the concept of division of labor, and today it is accepted almost as a truism. This linkage also underpins the concept of the experience curve, which posits that the marginal cost decreases while productivity increases with the cumulative level of output (Dutton and Thomas 1984). While many studies have established positive linkages between experience, learning, and productivity at an organizational level, scholars have noted that there are strong parallels between such learning at the organizational and individual levels (Argote 1993; Larson and Christensen 1993; Schilling et al. 2003).

Productivity gains due to experience accrue because tasks become more routine over time, and individuals gain greater tacit knowledge about the task (Argote 1999). The resulting learning curve at the individual level has been explained in terms of a three-fold transfer of learning: learning from past experience that is directly transferred to perform the task at hand at the same level as the last task completed, the application of learning from past tasks that is applied to make further adjustments in the way the current task is performed (thereby gaining some incremental productivity), and the application of the existing learning from previous tasks to improve the learning process from the current task (Ellis 1965).

In the context of software debugging, engineers gain both tacit and explicit knowledge with experience about the structure of the software code, linkages between different software modules, and the choice and sequence of debugging procedures to be followed in any specific context. They also learn to look for, and quickly identify important cues in the debugging process that reveal the nature and potential source of the bug. This knowledge, which is built up over the cumulative amount of time they spend at debugging tasks, enables them to shorten debugging times. Building on these arguments:

H1: Higher individual experience is associated with shorter debugging times.

Cumulative task variety

Individuals exposed to a variety of tasks can tackle problems within a single domain more efficiently and effectively. This positive effect of task variety can be manifested through multiple routes. First, exposure to task variety enables individuals gain knowledge about the broader schema that is relevant to each of the diverse tasks (Graydon and Griffin 1996; Paas and Van Merrenboer 1994). With this knowledge of the schema, the individual can better delineate knowledge that is relevant to the task at hand from knowledge that is less relevant. This prevents situations where the individual spends time and effort in mastering new knowledge that is not really useful to the current task.

Second, variety in tasks can also lead to implicit learning. While the individual may not even be aware of such learning, in the background, correlations between the task requirements in different domains are being stored in memory (Reber 1989; Simon 1991; Wulf and Schmidt 1997). This ultimately may lead to increased flexibility of the employee and contributes to better problem solving skills in any particular domain (Hopp and Oyen 2004; Hopp et al. 2004). Implicit learning is often reflected in tacit rather than explicit knowledge. For example, a youngster who has watched numerous basketball games may ultimately play basketball well and by the rules, but can be tongue-tied when asked to enumerate the rules of basketball. Whether the effects of exposure to task variety operate through the development of a more complete schema of relevant knowledge or of implicit knowledge, it has been argued that in either case "learning is transferred between related problem domains through the development of a deeper cognitive structure that applies to both…" (Schilling et al. 2003, p. 45). Finally, apart from enhanced learning, increased job variety has been associated with lower boredom and higher employee motivation, greater cooperation, better communication and consequently, higher productivity (Batt and Osterman 1993; Bishop and Kang 1996; Hopp and Oyen 2004).

These arguments are all applicable in the context of software debugging. First, because of the highly logical structure of software, working with a variety of software modules exposes the engineer to the various patterns in which blocks of software code are written, the ways in which blocks of software code interact within a module, and the ways in which software modules (which typically contain multiple blocks of software code) relate to one another. When an engineer knows more about this general schema that guides the design of the software, the engineer can often make more knowledgeable inferences regarding the potential source of the bug with limited examination of a specific module that is being debugged. Second, when software engineers work across modules, they develop numerous implicit rules regarding the classification of bugs and potential sources of those bugs and underlying common remedies. This knowledge is largely implicit and undocumented. In practice, many software engineers develop a reputation for being particularly effective in detecting and resolving certain kinds of problems, and their expertise is often drawn upon by their colleagues. Finally, as revealed in our field study and acknowledged by practitioners, software engineers quickly succumb to boredom with repetitive tasks and actively seek out variety both to relieve that boredom and to build a wider portfolio of marketable skills (Reed 2005). Building on these arguments:

H2: As the cumulative number of distinct software modules the engineer has worked with in the past increases, the average debugging time for allocated defects decreases.

Influence of task newness

While diversity in experience can enhance learning in the long run, individuals need time and effort to adapt to unfamiliar tasks (Edmondson et al. 2001). In the context of software maintenance, debugging an unfamiliar module may be challenging. Engineers spend time and effort to understand the code structure and domain of each module, and its linkages to related modules (Banker and Slaughter 1997; Littman et al. 1987; Rajlich 1999). Specifically in our study setting, classroom training played a minor role in preparing software engineers to tackle bugs in new modules. Instead, the engineers "self-trained" in one or both of the following ways. First, they worked through the code documentation and examined the code in detail to understand how the module worked and to what other modules it was linked. Second, they interacted with colleagues who had prior experience working on such modules. Accordingly, exposure to new variety can lead to serious short-term delays (Rus and Lindvall 2002). Building on these arguments:

H3: When engineers are allocated defects in software modules they are unfamiliar with, the debugging time for the allocated defects increases.

The transfer of experience to new tasks

Increase in experience improves skill levels of an individual, and results in enhanced productivity (Argote et al. 2003). Most tasks that individuals perform require learning more than one skill. For example, sending Morse code requires language, perceptual, and motor skills (Bryan and Harter 1899). While the time required to learn how to perform a task is proportional to the number of skills involved (Jovanovic and Griliches 1995), skills learnt can be used in other tasks that may be structured differently but call for the application of a subset of skills.

Second, when individuals work on distinct but related tasks, there emerges a kind of an "aha!" effect that characterizes insightful problem solving (Schilling et al. 2003). This happens because, with experience, individuals better recognize patterns among tasks that they encounter and develop deeper insights regarding those tasks compared to novices (Crossan et al. 1999; Fiol and Lyles 1985).

These arguments are applicable within the software maintenance context. Software debugging calls for multiple skills related to perception, cognition, and motor activity. These skills may be relevant, for example, to programming new code, understanding prewritten code, using equipment to replicate bugs, and using debugging and code commit tools. In this scenario, many of the basic skills gained from experience can be transported to new tasks that are encountered at work. For example, skills in using tools such as debuggers can be applied across software modules. In addition, as engineers gain experience at debugging, they are better able to understand the code flow and technology domain of any given software module, and the linkage of that module with other parts of the software. This understanding can help them cope better with newly allocated modules. Building on these arguments:

H4: The prior cumulative experience of the engineer moderates the effect of task newness on debugging time, in that engineers with greater cumulative experience require lesser time to debug unfamiliar modules compared to less experienced ones.

The tradeoff between exposure to variety and specialization

As argued earlier, exposure to task variety can enhance learning and productivity. However, too much variety may impede useful learning. In the manufacturing context, for example, a large variability in the options built into a car on the production line decreases productivity (Fisher and Ittner 1999).

In the context of task-related learning, the need for a balance between the exposure to task variety and specialization (as in repeated exposure to a certain task) can be motivated as follows.[9] First, exposure to variety may help individuals draw linkages between bodies of knowledge stored in long term memory, enabling them to better perform any given task. However, to be stored in long term memory, information has to first be stored and processed through the buffer of short-term memory, which is of limited capacity (Newell and Simon 1972). As noted by Simon (1990, p.2): "The number of chunks of information that can be held in short term memory is approximately seven." Further, repeated exposure to short term memory may be required for knowledge and skills associated with tasks to be correctly encoded into permanent memory. Thus, when an individual's exposure to variety is highly dispersed, there may not be a sufficient opportunity for the learning to be properly imbibed across tasks and then to be properly applied in a specific task.

Second, exposure to overly high variety may not allow enough time to put what knowledge is learned into practice. For example, when using a statistical package, an individual can follow and implement some code from a technical manual to estimate a model fitted to the data. However, for that knowledge to be efficiently applied, possibly with some adjustments, to estimate a differently structured model using a different data set, the individual must be able to put that knowledge into practice a few times and be comfortable with designing and implementing the code. That is, practice at various tasks is an important antecedent of effective learning from variety in a technically sophisticated setting.

Finally, as March (1991 p. 71) notes: "…maintaining an appropriate balance between exploration and exploitation is a primary factor in system survival and prosperity". While March's argument was made in the organizational context, its spirit also applies to individual

[9] An interesting parallel is that of preference learning of consumers. Hoeffler et al. (2006) suggest that the intensiveness and extensiveness of preference learning results in discovery of stable preferences.

learning. Overexposure to variety can lead to a lot of shallow learning, but insufficient application of that learning towards enhancing productivity.

These arguments all apply in the context of software debugging. In this technically sophisticated knowledge domain, software engineers who work across distinct modules can develop numerous implicit rules regarding the potential sources of those bugs and defect resolution approaches. For such learning to be ingrained in long-term memory and effectively transferred across tasks, however, engineers must have sufficient practice to understand the code base of each module. Such learning can be derived from repeated assignment to software defects within the same module. Accordingly, engineers would not benefit from fleeting exposure to debugging a variety of modules, but may need a certain level of immersion within each module for an enduring learning experience that can be applied elsewhere.

In addition, task variety can enhance motivation at work. Our field study revealed that engineers who worked on a variety of tasks tended to be highly motivated to learn from those tasks and apply that learning. This is consistent with the notion that software developers are inherently variety seeking (Reed 2005). On the other hand, managers we interviewed also emphasized that an engineer who worked on too many components generally felt overworked. These engineers were also called on more often to help their colleagues because their exposure to variety was seen as a signal of higher individual competency. These effects may act to lower motivation levels and productivity. Therefore, maintaining a right balance between specialization and variety of experience will likely maximize productivity. Building on these arguments:

H5: A balance between the breadth of experience gained due to exposure to a variety of software modules and the depth of experience gained by performing an adequate number of tasks within each module results in lowest average debugging times.

Team learning

The collocation of individuals in a team can enable greater learning and productivity. For example, engineers in a 'war room' setting typically display superior timeliness and productivity (Teasley et al. 2002). Productivity is higher within collocated teams because team members can learn from each other better, and because they can better coordinate their individual activities towards the shared overall goal. While team learning generally exerts a positive influence on productivity, such learning can arise out of "emotional algorithms" that may be of a competitive or a cooperative nature (Loch et al. 2006). On the competitive side, individuals may seek status within the group by sharing expertise or they might highlight their helping behavior with an objective of attaining some economic rewards. On the cooperative side, helping behaviors may result from the need for reciprocation (i.e., the need to return favors in kind) or from the need to identify more strongly with the group as a social level.

While positive implications of team-based learning and coordination have been established in domains as diverse as garment manufacturing and surgery (see Hamilton et al. 2003 and Reagans et al. 2005, respectively), these effects are particularly relevant in knowledge-intensive domains such as software maintenance. Software debugging is an unstructured task that involves substantial informal communication between team members. Such communication often happens during informal meetings (Brown and Duguid 1991). Further, software debugging tasks are highly situational, i.e., there is no exact pattern of bug resolution that works across all tasks. Such tasks are well-suited to collaborative learning (Tyre and von Hippel 1997).

Our field research revealed that the software engineers frequently interacted with each other on an informal level to solve problems. Most of the engineers working on a project were collocated. In that setting, we commonly encountered instances where engineers walked over to their more experienced colleagues to discuss issues related to the potential sources of bugs or to obtain their insights regarding the software code flow. The engineers also collaborated by using more formal mechanisms, including pre-scheduled presentation, and the use of team mail aliases. Intuitively, one would expect that as the number of collocated members in a team increase, the likelihood that the knowledge required to resolve a bug is both available within the team and is quickly applied to resolve the bug increases. Building on these arguments:

H6: As the size of the team an engineer works with increases, the average debugging time for the bugs allocated to that engineer decreases.

Entry and exit of team members

Churn in team membership can occur both on due to the exit of existing members and the entry of new members. The exit of a member results in the loss of collective organizational memory and can reduce productivity (Argote 1999; Johnson and Hasher 1987). In addition, when employees exit a team, issues related to task reallocation and workload realignment must be resolved – these adjustments take time in themselves, and also impose new learning needs on the team members.

Team member exit can have a particularly strong impact on productivity in the software context because a large fraction of the knowledge possessed by software engineers is tacit in nature. Such knowledge is difficult to code, and cannot be replaced easily or at short notice. At the same time, in large teams, engineers with overlapping sets of knowledge can step into the role of the departing member, and multiple team members can pool their expertise and energy to address the gaps caused by the departure with limited loss to their own productivities. Therefore, the negative effects of team member exit are more likely to be experienced in smaller teams. As demonstrated by Narayanan et al. (2006) in the software development context, the stability of smaller teams had a direct impact on the quality and timeliness of delivery in offshore projects – such stability mattered less in larger teams. Building on these arguments:

H7: As the ratio of the number of engineers exiting a team to the total number of team members increases in any time period, the average debugging time for the bugs assigned to an engineer in that team increases.

While discussion on team turnover both in practice and the literature focus mainly on exit, the entry of new members can also reduce productivity.[10] This is consistent with the well-known Brook's Law: "Adding manpower to a late software project makes it later" (Brooks 1995, p. 25). New engineers not only need to learn about their tasks, but also about the communication patterns, cultural practices, and inter-individual relationships that guide

[10] See Carley (1992) for a simulation study of the impact of both team joining and exit.

the working of the team. Existing team members may invest significant time and effort to facilitate such learning on the part of incoming team members (Mincer 1962).[11]

In our study context, engineers joining a team frequently disrupted the work of existing members to seek their insights regarding both the organizational setup, and technical issues related to bug reproduction, bug fixing, verification of the fix, and finally committing the corrected code. While team members would cheerfully extend their cooperation in most cases, such effort would likely lower their own productivity. Further, this negative impact on productivity is likely to be higher for smaller teams where the burden of training a new individual cannot be shared across numerous team members. Building on these arguments:

H8: As the ratio of number of engineers joining a team to the total number of team members increases in any time period, the average debugging time for the bugs assigned to an engineer in that team increases.

Control variables

We control for multiple variables that are of lesser theoretical interest but may also impact productivity in a software maintenance environment. First, we control for the work-in-process, or the total number of assigned bugs to an engineer but unresolved at any given point in time. A high work-in-process can enhance productivity (because of the pressure to get the job done and move along) or detract from productivity (because it may reduce the engineer's focus and concentration). Second, we control for bug severity. This is an urgency measure allocated by the supervising manager that reflects the degree to which the bug impairs overall software functionality (and hence needs urgent attention). Third, we control

[11] Apart from separately studying the impact of joining and exit, we also study the impact of '*cumulative turnover*' (defined as the sum of new joining and employee exits) on productivity.

for total days of training the engineer received, including (a) technical training (focused on tool usage, programming skills, and domain knowledge), (b) process and quality training, and (c) communication and cultural training. Fourth, we control for whether the engineer seeks additional information from the person who filed the bug to assist the engineer in defect replication and resolution. Anecdotal evidence suggests that during debugging, significant time is taken to just replicate the bug in the software. This requirement of additional information is well-recognized in models of software debugging (Hale and Haworth 1991). Operationally, the bug is effectively in "sleep" mode while information is awaited – hence the importance of this control.

Finally, the following variables, which could be candidates for additional control variables, are excluded from our model. First, we considered the "defect-reopening-rate" which measures the percentage of defects reported as resolved by an engineer during a time period that are re-reported as bugs at a later point in time. That is, an engineer trying to boost his or her reported bug-fixing productivity by delivering a number of hurried, but flawed bug fixes. However, we found that less than 1.5% of all reported bug fixes were ever re-opened. Second, the productivity of engineers could vary as a function of the set of available software analysis and debugging tools. However, all engineers we studied worked with the same set of tools. Finally, debugging rates in software projects may be related to the overall type of software a team is working on. In our dataset, the teams were involved in working on different modules of the same large piece of system software, used the same programming language, and applied broadly similar skill sets.

3.3. Data, measures and model

Each engineer we study belonged to a team that was responsible for the corrective maintenance of one or more modules of a large system software configuration, where a module is defined as a piece of software code that has a distinct, well-defined functionality and is a part of larger body of software. Our dataset included details regarding the bug history, the engineer who performed the debugging, the date that task was assigned, bug severity, whether and when the engineer asked for additional information, the module in which the bug originated, and finally, the bug resolution date. The resolution process of the bug is as follows: A bug's designation is changed from "*open*" to "*assigned*" when it is assigned to an engineer. If all the information is not available, then the engineer seeks more information on debugging. If all information is available, an engineer works on the bug and at some date marks the bug as resolved along with resolution comments that indicate whether the bug is resolved[12], a duplicate[13], not reproducible[14] (after a few rounds of trials), and closed[15] (reasons are outlined for this state, which is reached only after significant effort has been expended and if all the parties agree that it is reasonable to do so). We also collected data from the human resources department on the training provided to the engineer, the date of joining the firm, the team the engineer belonged to, and the date the engineer moved out of the team.

All the engineers working on a project were co-located, and interacted both in person and via email. The engineers possessed broadly the same set of skills ('C', Unix) but the

[12] The bug has been resolved and fixed

[13] The bug report describes the same problem as in another report

[14] Problem cannot be reproduced by the evaluating or the test engineer

software modules worked on could vary across projects. An engineer could be involved in debugging more than one module during any time period, and could over a period of time be assigned to debugging tasks in new modules that he/she had not worked on earlier. The manager in charge of the team was responsible for task assignment. Our final dataset consisted of 12503 valid bugs handled by 193 individuals from 31 teams.

Measures

Dependent variable (M_{it})**:** Our dependent variable is the average time required by engineer '*i'* to resolve defects that are assigned in month '*t'*. This measure has been used earlier (Huntley 2003), and is consistent with the productivity measure used by the firm (Mean Time to Resolve bugs or "MTTR" in a given month).

Independent variables

Individual Experience (Log(EXP_{it})): This is the natural log of the total number of days (*EXP*) since the engineer *i* joined the company as of the end of a given month *'t'*.

Variety or unique module allocation (I_{it})*:* We assigned a categorical variable (I_{it}=1) for each module that the engineer had not worked on at any point in the past, but that was assigned to engineer '*i*' during time *t*. $I_{it} = 0$ indicates that no new module was assigned to individual '*i*' at time '*t'*.

Cumulative variety (V_{it-1})**:** The total number of unique modules allocated to an individual from time 0 through '*t'*, captured by $V_{it} = \sum_{s=0}^{t} I_{is}$.

[15] A bug report is valid, but a conscious decision has been made not to fix the bug. The rights to this are available to the component owner.

Team Size (*TEAMSIZE*)**:** The total number of engineers in the team *'z'* of engineer '*i*' at time '*t*'.

New Joining and Exit (*NEWJOINING/TEAMSIZE and EXIT/TEAMSIZE*)**:** New Joining (exit) is represented by the total number of engineers who joined (left) the team '*z*' of engineer '*i*' during time period '*t*'. Over the time period we study, 119 engineers joined various teams and 48 engineers departed from their teams. In most cases of exit, a replacement was inducted into the team. In operational terms, we use the ratio of engineers joining or leaving the team to the total number of engineers in the team at time *t* (including those who left and joined during the period).

Herfindahl-Hirschman Experience Index **(***HHEI***):** We encountered an interesting challenge when measuring the variety of experience. Specifically, the number of unique modules an engineer worked on is not, in itself, a good measure of variety from a learning perspective. For example, consider an engineer with a total experience of 15 bugs who was exposed to 5 distinct modules and resolved 3 bugs on each module. In contrast, consider an engineer with the same total experience who has resolved 11 bugs on one module and one bug on each of 4 other distinct modules. In line with the theoretical arguments presented earlier, the first engineer presumably has a superior exposure to variety from a learning perspective compared to the latter. We encountered many such instances of variation in exposure to variety across the engineers in our data.

To address this issue, we adapt and apply the Herfindahl-Hirschman Index (*HHI*), which has been used to measure market concentration in economic and antitrust analyses. This index varies between 1/*N* and 1, where *N* is the total number of firms in the market. When market share is evenly dispersed across the *N* firms, *HHI*=1/*N*. In contrast, *HHI*=1 when one firm has all of the market and the other firms have infinitesimally small market shares – this essentially represents a monopoly.

We use the following adaptation of the *HHI* – termed the Herfindahl-Hirschman Experience Index (*HHEI*) henceforth – to measure the dispersion of an engineer's exposure to distinct modules:

Let C_{ikt} : Total bugs handled by engineer '*i*' on module '*k*' until time '*t*'

D_{it} : Total number of bugs handled by engineer '*i*' in time '*t*' across all modules he or she has worked on until time '*t*'

$P_{ikt} = \frac{C_{ikt}}{D_{it}}$: Cumulative proportion of bugs worked on in module '*k*' by individual '*i*' until time '*t*'.

Then $HHEI_{it} = \sum_{\forall k} P_{ikt}^2$ is the Herfindhal-Hirschman Experience Index of engineer '*i*' at time '*t*' based on all components *'k'* that are worked on until time *'t'*.

Note that the *HHEI* for an engineer will change across time period to accommodate the changing allocation of tasks over time. A *HHEI* of 1 indicates that the engineer has worked on just one module, i.e., the experience is highly concentrated. In contrast, when *HHEI* is at its minimum possible value of 1/*N*, the engineer's experience is equally dispersed across the modules he or she has worked on, with no single module responsible for a large fraction of the experience. Continuing with our example at the beginning of this section, the first engineer who has worked on 3 bugs in each of 5 modules will have a *HHEI* of 0.2 (0.2^2+0.2^2+0.2^2+0.2^2+0.2^2 = 0.2), reflecting the dispersed nature of experience. However, the second engineer would have a *HHEI* of .556 that reflects the greater concentration of experience in one module.

Control Variables

***Work-in-process** (W_it)*: This is the total number of defects that were left over unresolved in period '*t*' that the individual would have to carry forward into the next period. This is given by $W_{it} = W_{it-1} + A_{it} - R_{it}$ where $W_{i(t-1)}$ is the Work-in-progress in period 't-1', A_{it} is the number of bugs assigned in period '*t'* and R_{it} is the number of bugs resolved in period t.

***Severity** (S_{it})*: Severity ratings varied from 1 to 6. A Severity rating of 1 indicated that the bug did not impair functionality of the software. In contrast, a severity rating of 6 indicated that the bug impacted basic software functionality – hence early resolution was important.

***Training** ($T_{it:}$)*: This is the total number of days the individual spent in various training programs.

***Fraction Information requirement** (INF_{it})*: This is the fraction of all assigned defects during any time period '*t'* that required additional information at some stage before the bug was ultimately resolved.

Summary statistics for the variables are in Table 11, and bivariate correlations are in Table 12.

3.4. Model, Estimation, and Findings

We test our hypotheses using the following specification, termed Model 1:

$$\begin{aligned}
M_{it} = \alpha + \beta_1 \operatorname{Log}(EXP_{i(t-1)}) + \beta_2 I_{i(t-1)} + \beta_3 I_{i(t-1)} * \operatorname{Log}(EXP_{i(t-1)}) + &\left.\right\} \text{Experience and Unique Module} \\
\beta_4 V_{i(t-1)} + \beta_5 HHEI_{i(t-1)} + \beta_6 HHEI^2_{i(t-1)} + &\left.\right\} \text{Variety and Herfindahl-Hirschman Experience Index} \\
\beta_7 (TEAMSIZE) + \beta_8 (NEWJOINING / TEAMSIZE)_{iz(t-1)} + \beta_9 (EXIT / TEAMSIZE)_{iz(t-1)} + &\left.\right\} \text{Team Size and Turnover} \\
\beta_{10}(INF_{it}) + \beta_{11} S_{it} + \beta_{12} T_{i(t-1)} + \beta_{13} W_{it} + \varepsilon_{it} &\left.\right\} \text{Control variables}
\end{aligned}$$

.... MODEL 1

Variable	Minimum	Maximum	Global Average	Standard Deviation
Individual Experience (days in Project)	79	2907	1093.3	754.14
Number of different Modules Handled	1	30	7.98	5.65
Team size to which an Individual Belongs	1	13	7.16	2.86
Total Number of Bugs Handled	2	440	64.7	70.34
HHEI Summary of Individual Averages	0.11	1	0.56	0.26
Summary of Work in Process means for each Individual	0	8.33	3.08	2.71
Severity of Bugs	1	6	3.07	0.79
Panel Length for an individual	2	65	15.97	11.81
Average Time to resolve bugs in days	1	903	48.43	79.29
Average resignation/month across projects over the time span of data collection for the project	0	2	1.06	0.574
Average Joining/month across projects over the time span of data collection	1	2	1.19	0.275

Table 11 # Summary statistics for the panel data

		(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
Average time to Resolve	**(a)**	1									
Individual experience (lag)	**(b)**	-0.064	1								
Team Size	**(c)**	-0.127	-0.038	1							
Variety Experience (Lag)	**(d)**	-0.129	0.318	0.03	1						
Work in Progress	**(e)**	0.273	0.088	-0.149	0.154	1					
Average Severity	**(f)**	-0.128	0.096	-0.008	0.145	0.027	1				
Fraction Requiring Information	**(g)**	-0.027	0.071	0.085	0.066	-0.038	0.255	1			
Ratio of Engineers Joined to Team size (Lag)	**(h)**	0.047	-0.026	0.027	-0.032	0.003	-0.014	-0.017	1		
Ratio of Engineers Resigned to Team size (Lag)	**(i)**	0.009	0.017	0.002	0.037	0.01	-0.002	0	0.027	1	
Allocation of Unique Components (Lag)	**(j)**	0.061	-0.224	-0.054	0.033	-0.007	-0.042	-0.018	0.02	0.016	1
HHEI	**(k)**	0.132	-0.194	-0.116	-0.592	-0.034	-0.089	-0.067	0.013	-0.005	-0.158

Table 12 # Bivariate correlations between the variables

3.5. Estimation

We estimated our model using a Fixed Effects regression for the following reasons. First, the R-square for a model with fixed effects for engineers was significantly higher than that for the pooled OLS (the results of the F-test for the significance of the fixed effects coefficients are in Table 12; p-value = .0001). Second, a Hausman test of the difference between the coefficients of the fixed and random effects models suggested the presence of possible endogeneity (Hausman 1978). A fixed effects specification accounts for endogeneity by accommodating the correlation between the independent variables and the error term.

In panel data sets, errors across time periods for an individual may be correlated when the panel contains lengthy sequences of data for individuals (Petersen 2005). This leads to a downward bias in the estimates of the standard errors, resulting in higher t-values. This bias in standard error can be adjusted for by using cluster correlation to yield unbiased estimates of standard errors (Wooldridge 2002; Petersen 2005). Therefore, we corrected for possible auto-correlation using the clustering option in STATA (Rogers 1993).

3.6. Findings

Estimation results are in Table 13. First, cumulative experience reduces debugging time (supporting Hypotheses H1). Controlling for cumulative experience, we find that exposure to variety of experience (or cumulative task variety) is associated with lower debugging times (supporting Hypothesis H2). In the context of task newness, we find not only that debugging time increases as engineers encounter new modules (supporting Hypothesis H3), but also that this impact of newness is reduced when the engineers are more experienced (supporting Hypothesis H4). This suggests that experience in debugging existing modules is transferred to debugging unfamiliar modules. Note that exposure to new modules is what leads to enhanced variety of experience, which, according to Hypothesis H2, leads to a decrease in debugging times. To reconcile Hypotheses H1 and H4, it is important to contrast the short-term and long-term effects of exposure to variety. The subtle but important distinction we find here is that, while cumulative exposure to variety over time improves productivity, exposure to variety involves some short-term productivity loss as engineers acclimatize to, and learn about, the new task. (also see Carillo and Gaimon 2000).

Importantly, we find support for the argument that there is a tradeoff between experience gained due to exposure to a variety of software modules and the repeated allocation of bugs in a given module (specialization), and that the two need to be balanced to maximize learning and productivity. After including both the cumulative experience and cumulative exposure to variety (i.e, the number of distinct modules worked on) in the model (see Table 13), we find that the linear coefficient of HHEI is significantly negative whereas the non-linear coefficient is significantly positive (supporting the non-linear Hypothesis H5). Intuitively, for very low values of HHEI, experience tends to be highly dispersed. In this case, focusing the engineers on fewer modules and increasing the specialization on each module (which increases HHEI), will lead to lower debugging times and increase productivity. In contrast, for high values of HHEI, the experience of the engineer is already highly concentrated on a few modules. In this case, adding variety to an individual's experience will reduce HHEI, thereby reducing debugging times and increasing productivity. While the role of specialization in learning has been well-established, the role of variety has been addressed in fewer and more recent contexts (e.g., Schilling et al. 2003). Our work adds to existing insights in the area by highlighting some of the short-term negative effects of exposure to variety and by empirically establishing the need for a balance between exposure to variety and specialization.

Moving to issues of team size and churn, we find that larger team sizes are associated with shorter resolution times (supporting Hypothesis H6). In our study setting, there was a free flow of information between team members. There were no relative performance measurement systems implemented, and the organizational culture was one of cooperation and openness. Arguably, the larger inventory of skills and experience available within larger teams helped increase productivity. In terms of turnover, we find no evidence to support the notion that the debugging time increases as the ratio of engineers leaving a team to the total team size strength increases (i.e., Hypothesis H7 was not supported). We find, though, that debugging time increases when the ratio of engineers joining a team to the total team size strength increases (i.e., Hypothesis H8 is supported). We also estimated a model where we added new joining and exit to create a single variable that measured "team turnover". This variable (Exit/ team size), when included in the regression in place of the new hire and exit ratio variables, also significantly impacted debugging times.

These results add to existing perspectives on turnover and learning in the literature. First, some studies – for example, the study of the shipbuilding industry by Argote et al. (1990) – have found no evidence that turnover significantly impacted learning and productivity. However, this impact of turnover may be sensitive to the work context (Argote 1999). In the context of software debugging, the considerable lead time involved in learning can impact productivity in case of turnover. Second, our analysis suggests that turnover may best be measured not in absolute terms but as a fraction of team size. This approach allows us to implicitly focus on the percentage of knowledge turned over. Finally, we note that our finding that engineer exit does not significantly impact productivity may reflect the fact that the team usually has some time to prepare for exit once engineers give notice of their intentions to leave the team or the company. For example, other engineers may gradually ease into new tasks and roles in preparation for the exit. On the other hand, relatively little can be done, and little is usually done in the field, to prepare for the entry of a new team member. This is because the kind and level of attention the new entrant requires from existing members is largely unknown and can vary strongly across entrants.

Finally, for the control variables we find that, first, the severity of the bug is negatively associated, and work-in-process (WIP) is positively associated, with debugging time. Interestingly, the training provided to engineers did not have a significant impact on debugging time. We also estimated three additional models where the total number of days of training was replaced by the total number of days related to each of the three sub-types of training discussed earlier. None of the sub-types of training significantly impacted debugging time. Ex-post discussions with the firm's managers revealed that the training offered was generic in nature and primarily aimed at familiarizing the engineers with the "language" of the workplace. Consistent with the observations of Rus and Lindvall (2002), the managers believed that most of the training that impacted productivity happened on the job.

Variables	**Coefficient**	**Robust Standard Error**	**P Value**
Log Individual Experience (t-1)	-0.102	0.058	0.082
Total Team size	-0.044	0.021	0.039
Cumulative Variety Experience (t-1)	-0.032	0.013	0.017
Work in Progress	0.097	0.008	0.000
Average Severity	-0.203	0.04	0.000
Fraction Requiring Information	0.07	0.072	0.334
Joined/Team size at (t-1)	0.36	0.165	0.030
Turnover/Team size at (t-1)	0.409	0.293	0.165
Unique module allocation (t-1)	0.323	0.149	0.032
Interaction between unique module allocation and experience (t-1)	-0.046	0.024	0.053
Herfindahl-Hirschman Experience Index (t-1)	-1.427	0.771	0.065
Herfindahl-Hirschman Experience Index Square (t-1)	1.406	0.609	0.022
Constant	4.682	0.391	0
Adjusted R-Square	0.2691		
Mean Square Error	1.0162		
N	2794		
F test for Engineer Fixed effects	2.287		0.0001

Table 13 # Estimates calculated for Model 1

Robustness checks

As a first step, we checked for the robustness of our findings by dropping selected variables to observe any changes in the coefficients of the remaining variables. Our estimates were stable to this manipulation. We first dropped the variables related to unique experience and the interaction of unique experience with cumulative experience. This did not vary the coefficients of *HHEI* and *HHEI*^{2} significantly, and other coefficients also remained unaffected. Similarly, dropping *HHEI* and *HHEI*^{2} did not impact other coefficients significantly. However, our analysis revealed that when *HHEI*^{2} alone was dropped, the sign of the linear coefficient of HHEI flipped from negative to positive but was not significantly different from zero (prob = 0.13). Figure 2 captures the underlying intuition. The non-linear effect in the figure is correctly picked up in Table 13 by the negative linear coefficient and the positive quadratic coefficient. However, when the quadratic coefficient is absent, the linear coefficient (slope of the dashed line in Figure 2) weakly picks up the increase in debugging time for high levels of *HHEI* (the right-hand side of Figure 2).

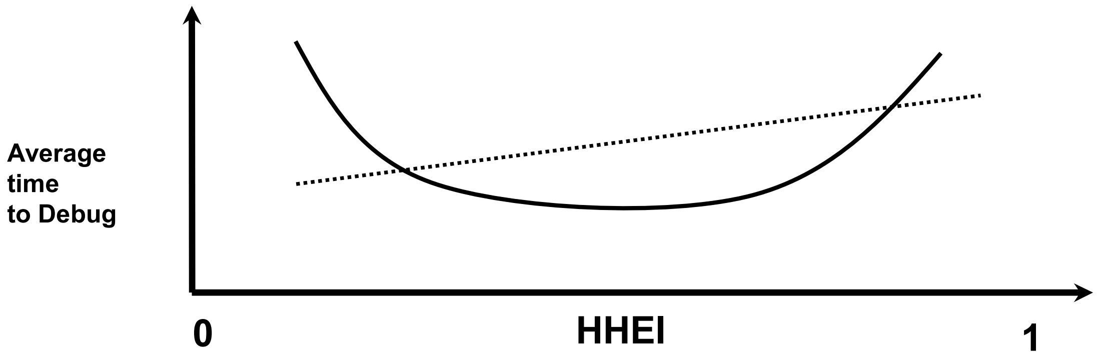

Figure 2 # Illustration of HHEI – Linear and Nonlinear effects

Second, to better understand the impact of variety on productivity at different levels of HHEI and to further validate the nonlinear impact of HHEI, we categorized HHEI based on the median observation (HHEI =0.483), and estimated a related model (Model 2) described below:

$$\left.\begin{aligned} M_{it} = \alpha + \beta_1 \operatorname{Log}(EXP_{i(t-1)}) + \beta_2 I_{i(t-1)} + \beta_3 I_{i(t-1)} * \operatorname{Log}(EXP_{i(t-1)}) + \end{aligned}\right\} \textbf{Experience and Unique Module}$$

$$\left.\begin{aligned} \beta_4 V_{i(t-1)} + \beta_5 CATHHEI_{i(t-1)} + \beta_6 (CATHHEI * V)_{i(t-1)} + \end{aligned}\right\} \textbf{Variety and Herfindahl-Hirschman Experience Index}$$

$$\left.\begin{aligned} \beta_7 (TEAMSIZE) + \beta_8 (NEWJOINING / TEAMSIZE)_{iz(t-1)} + \beta_9 (EXIT / TEAMSIZE)_{iz(t-1)} + \end{aligned}\right\} \textbf{Team Size and Turnover}$$

$$\left.\begin{aligned} \beta_{10}(INF_{it}) + \beta_{11} S_{it} + \beta_{12} T_{i(t-1)} + \beta_{13} W_{it} + \varepsilon_{it} \end{aligned}\right\} \textbf{Control variables}$$

.... MODEL 2

Here *CATHHEI* is a categorical variable with 0 (1) indicating Low (High) *HHEI* for the engineer in a time period. Specifically, engineers with a *HHEI* during a time period of greater (lesser) than 0.483 were coded 1 (0). Note that in this model *CATHHEI* is allowed to interact with cumulative variety $V_{it.}$ The estimates of this model are presented in Table 13. These estimates are generally consistent with those in Table 3. More importantly, this model helps us validate the nonlinear effect of the interaction between *HHEI* and its interaction with variety. The estimates indicate that when *HHEI* is high (=1), i.e., the past experience is highly concentrated, an increase in cumulative variety reduces debugging time and improves productivity. When *CATHHEI* =1, note that the total coefficient for cumulative variety in Model 2 is $(\beta_4 + \beta_6)$. The estimate for this coefficient was -0.078, which is significantly different from zero (p = 0.000, *t* = -4.93, 95% confidence interval of -109 to -.046). Further, when *HHEI* is set to zero, the coefficient of variety (now β_4) is not significantly different from zero. This suggests that increasing variety may help increase productivity when the experience of an individual is highly concentrated, but not when the experience of the individual is already highly dispersed. This argument is consistent with, and increases our confidence in the insights obtained from the estimation of Model 1.

Variable	Estimates	Robust Std. error	P-Value
Log Individual Experience (t-1)	-0.11	0.055	0.048
Work in Progress	0.098	0.008	0.000
Average Severity	-0.205	0.04	0.000
Fraction Requiring Information	0.072	0.073	0.325
Total Team size	-0.042	0.020	0.043
Cumulative Variety Experience (t-1)	-0.019	0.013	0.163
Unique module allocation (t-1)	0.287	0.141	0.043
Interaction between unique module allocation and experience (t-1)	-0.04	0.022	0.076
Joined/Team size at (t-1)	0.351	0.163	0.032
Turnover/Team size at (t-1)	0.475	0.271	0.081
Interaction of Categorical HHEI and Cumulative variety experience (t-1)	-0.059	0.015	0.000
HHEI Categorical (1 is High and 0 is low) (t-1)	0.36	0.131	0.007
Constant	3.745	0.349	0.000
Adjusted R-Square	0.267		
Mean Square Error	1.017		
N	2794		
F test for Engineer Fixed effects	2.295		0.000

Table 14 # Estimates for Model 2

3.7. Conclusion

Our findings have implications for both managers and researchers.

Managerial Implications

First, from the perspective of learning patterns, our findings suggest that managers must strive for the correct balance between specialization and exposure to variety in training. In fact, in our study, the effects of initial training were not significant in enhancing productivity (although some of that training may be necessary to “break-in” the engineers and get them started on the job). Much if the learning happens on the job. Therefore, managers must carefully consider the stage in which correct balance between specialization and exposure to variety is achieved, and its variation across individuals.

Second, managers must be sensitive to the notion that there may be “too much variety.” In this context, our empirical findings reveal that learning and productivity drop when experience is overly scattered across different modules. In addition, our field research indicated that engineers who handled an excessively large variety of modules felt that they were penalized more for being productive.

Third, our findings indicate that exposure to variety reduces productivity in the short term. Therefore, managers must both schedule an engineer’s exposure to variety when peak productivity is not required and must be prepared to wait awhile before the gains from variety-driven learning are realized.

Fourth, managers must carefully consider how tasks must be allocated across the members in a team so that the overall productivity of the team is maximized. This decision has both spatial (task allocation across team members at a point in time) and temporal (change in the allocation of tasks over time) dimensions. For example, in the short term, our findings indicate that the most experienced member of the team will be the most productive at a new task. However, for long term productivity, it may be optimal to allocate that task to a less experienced team member who is in a position to learn the most from such exposure. This member may struggle with the task, but the ultimate gain in long term productivity will likely be the highest in his or her case. A possible allocation heuristic could involve exposing new members to a few modules to increase exposure to variety and to enhance rapid learning, and then introduce occasional variety possibly to enhance their intuitions on existing tasks as they gain in experience.

Finally, managers may need to shift from thinking about team turnover as a single concept, and focus on the separate effects of entry and exit of individuals into and from the team. While managers are generally concerned about exit, our findings suggest that entry may be more of a problem at least in the context of short term productivity. In most firms, an employee is required to provide notice sufficiently in advance of exit – this allows the work transition to be carefully planned so that productivity is maintained. However, the entry of a new employee is more difficult to plan for because the kind and depth of resources he or she will draw from the remaining team members to get up to speed are difficult to predict.

Theoretical Implications

Whereas the benefits of specialization have been expounded since Adam Smith, the role of task variety in enhancing learning has been recognized, or at least, sufficiently highlighted, only in relatively recent times (e.g., Schilling et al. 2003). Our work adds to existing perspectives on task specialization and variety by theoretically motivating and empirically establishing the idea of learning by doing in a real-life, knowledge-intensive, while collar environment. In particular we find that the best learning occurs when there is a certain balance between specialization and variety.

Second, while our analysis was mainly focused on individual learning, the results have implications for group/team learning as well. This is consistent with the notion that, depending on the context, the concepts of individual, group, and organizational learning may be highly interconnected (Argote 1993, Larson and Christensen 1993). In the firm we study, learning occurs not just from exposure to variety and specializing in tasks but also from the sharing of knowledge between team members. Therefore, if the individual is trained well, there are potentially some ripple effects on learning and productivity at the group level as well (Siemsen at al. 2006). Additionally, the firm usually benefits from overall team-level productivity more than it benefits from the productivity of an individual within the team.

Third, in the methodological context, our conceptualization of the Herfindahl-Hirschman Experience Index (HHEI) can be applies to measure the dispersion of experience in future empirical studies of learning and associated phenomena.

Finally, our findings on the team size, new joining and employee exit suggest that additions to and attritions from teams have distinct impacts on learning and productivity. Hence research must ideally consider and evaluate these concepts separately. In addition, we find that the absolute number of new or departing team members is less significant than the fraction of members joining anew or departing from the team. This is because the variables of greatest operational relevance to overall team productivity are the percentage of knowledge turned over (when team members depart) and the fractional additional load imposed on an existing team member (in case incoming team members need to be brought up to speed).

CHAPTER 4

4. Optimal Resource Allocation in Software Maintenance

4.1. Introduction

A recent report by NIST (2002) estimates that the annual cost of software bugs[16] to the economy is a staggering \$59.5 billion. The key reasons cited are (1) lack of testing infrastructure; (2) inefficiency of testing process in capturing bugs and (3) progressively increasing complexity of software. It is acknowledged in the software maintenance literature that perfect software with zero bugs is unrealistic due to continually evolving nature of the product (Liberman 1997). Further, software maintenance is costly – maintenance budgets constitute between 50% and 80% of the overall IS budget (Nosek and Palvia 1990), and costs of maintenance are known to outweigh the costs of development by a factor between 2 and 10 (Scacchi 1994). A considerable amount of the expense is expended on labor – close to 75% (Amoribieta et al. 2001). Because the current business environment is characterized by scarce labor resources and high employee turnover– only 25% of hired engineers continue with a company for more than 5 years (Sudhakar 2002), managers handling maintenance activities are challenged to utilize available resources effectively.

Increase in utilization of engineering resources can be accomplished in two ways. First, the productivity of engineers can be increased by increasing the rate of learning of

[16] Problem due to which the software does not meet the intended specifications

employees. This can be done by allocating tasks to individuals based on the amount of variety in their current assignments and the concentration of experience (Narayanan et al. 2006), collocating team members (Teasley et al. 2002; Narayanan et al. 2006) and effectively managing the process of employee entry and employee exit (Narayanan et al. 2006). Second, adjustments can be made to the process of debugging based on the properties of the debugging task – focus of the current work.

Software maintenance is divided into perfective, adaptive, corrective and preventive maintenance (Leintz et al. 1978). Corrective maintenance involves fixing bugs in the software. Perfective, adaptive and preventive maintenance tasks may involve addition of new features to improve functionality of the software or modifying the software specifications to meet future needs. A considerable amount of current academic work in software maintenance – from a management science perspective – is oriented towards understanding software enhancement (e.g. Banker and Slaughter 1997; Banker et al. 1998; Krishnan et al. 2004). At the bug level, considerable work has been done on predicting occurrence rates of bugs (e.g. Elrich and Emerson 1987; Goel and Okumoto 1979; Littlewood 1980; van Pul 1994).). Despite considerable amount of research on software maintenance, investigation on debugging process has been scarce to the extent that it has been labeled as a "scandal" by Liberman (1997). Current research in software debugging has focused on understanding cognitive behavior of engineers undertaking the debugging task (Hale and Haworth 1991; Hale et al. 1999). However, there is little or no work to the best of our knowledge that offers a management science perspective to the debugging process. Further, management of debugging efforts – queue management – is different from the management of enhancement activities in software – project management (April et al. 2004). Our work differs from the prior literature in software maintenance in the following aspects.

First, we focus on individual software debugging tasks unlike enhancement projects as in the past literature (Banker et al. 2002; 1991;1998; Banker and Slaughter 1997; Kemerer and Slaughter 1997; Krishnan et al. 2004). Next, our focus is on the resources – labor resources – and utilization of those resources to extract maximum efficiency and not the software per se. Finally, we focus on capacity planning and enumerate the varied tradeoffs that managers face between the choice of capacity, queue length and management of the diversity of incoming tasks when resource endowments are limited.

Our decision model involves two stages. In the first stage, we draw from theory and propose alternative econometric models of bug resolution. These models are then estimated and validated using bug resolution data from a large software software company. Based on this analysis, we draw the insight that the fraction of bugs successfully resolved in each period decreases, as the bugs stay in the system longer. The best fitting model – which specifies a beta geometric distribution of the conditional probability of bug resolution during a given time period – is taken forward to the second stage.

In the second stage, we develop a heuristic estimate of the effort required to resolve the incoming bugs under various output requirements. We present an optimization algorithm that imposes cut-off policies based on the time that bugs should be worked on without finding successful resolution. We demonstrate that such policies minimally impact rates of successful bug resolution, while simultaneously reducing waiting times for the assignment of incoming bugs to engineers and improving the overall productivity of the maintenance operation. We extend our analysis to consider resource allocation decisions when the resource allocation priority can vary across the categories of incoming bugs.

In § 2, we describe the research setting and the dataset used. In § 3, we examine competing model specifications and choose the model that provides the best prediction of the resolution phenomenon observed in our data. In § 4, we examine model validity and understand the implications of imposing a policy of threshold time period to cut off working on bugs. In § 5, we extend our analysis of the threshold policies to multiple groups. In § 6, we use queueing approximations from the literature to analyze the tradeoff between imposition of threshold policies and the waiting time of bugs in the system for two different queueing systems – First Come First Serve (FCFS) and Pre-Emptive Resume priority. Finally, in § 7, we outline limitations of the model, discuss the managerial implications and possible generalizations.

4.2. Research setting and data

Research Setting

The data for the study was sourced from a large India-based, export-oriented software services provider. The firm employs over 10,000 engineers and develops and maintains both application and system software. Exports comprise more than 90% of the firm's revenue. The research effort included on-site field work conducted over two months at the firm's operating sites in India. The data covers 12503 debugging tasks performed on bugs originating from a large piece of system software. These tasks were assigned to engineers in project teams. The teams were supervised by lead engineers.

Resolution Process

Each incoming bug is assigned to an engineer who works on the bug individually. Engineers also seek opinion of their counterparts or solicit help from email listserve's and product documentation. Figure 3 shows the process of bug resolution. The resolution process is as follows: (1) the engineer investigates whether the bug is *valid*. If the bug is not *valid* bug then it can be designated as *junk*[17]. If the bug is valid, then the engineer tries to reproduce the bug in a laboratory setting. (2) If the bug is reproduced, then an attempt is made to resolve the bug. If not, the attempts to reproduce the bugs continue. If attempts to reproduce are repeated unsuccessfully, bugs can be designated as *irreproducible*[18] (3) If the bug is successfully reproduced, an engineer attempts to resolve the bug. Such attempts can result in the bug being successfully resolved (Either *resolved*[19] or *duplicate*[20]) or not being successfully resolved resulting in a *closure*[21]. We group *irreproducible* and *closed* bugs and classify them as *unsuccessful resolutions*. These are *resolutions* because effort is invested in resolving such bugs leading to an end state. They are *unsuccessful* because they do not directly contribute to improvement in the overall quality of the product. The other bugs are

[17] The bug does not require any changes to the software

[18] Problem cannot be reproduced by the evaluating or the test engineer. Typical reasons for a bug to be declared irreproducible bug is "*compatibility problems with different software*."

[19] The bug has been resolved and fixed

[20] The bug report describes the same problem contained in another report

[21] The bug report is valid, but a conscious decision has been made not to fix the bug. The rights to this are available to the component owner. Typical reasons for *closure* of a bug are (a) a one off case that is not expected to repeat because the customer used the software in conditions that were not in conformance to specification (b) technical changes needed to fix the bug could potentially create problems in other features of the software.

designated as *successful resolutions*. They are s*uccessful* because the bug has either been fixed or is tracked and tagged to a *known* problem.

Our field interviews with the managers suggested that the engineers tend to work on problems in an iterative manner. On occasions, when a clear solution is not available, the engineer rests on the problem before making another attempt on the problem. This is in line with other investigations of engineer behavior during the debugging process (Hale and Haworth 1991). In any resolution iteration, the engineer generates a hypothesis on the cause of the bug, and implements a resolution tactic based on the current hypothesis. If the hypothesis is correct, the cause of the bug is found and the bug is resolved. If the hypothesis is incorrect, a different hypothesis is generated and another attempt is made at the resolution (Hale and Haworth 1991).

Table 15 presents the resolution data with the time span of resolution divided into months – henceforth called a *period*. As seen in Table 15, in each period, some bugs are successfully resolved, some are designated as *closed* or *irreproducible* and the remaining bugs are carried forward into the next period since no conclusion can be reached on their status. As seen in Table 15 (Figure 4), the proportion of successful resolutions decreases with time. The intuition behind this decrease can be explained based on the Hale and Haworth (1991) model of debugging mentioned above. With each unsuccessful hypothesis generation exercise, the engineer takes longer to generate the next hypotheses given limitations of current know how. Also, the number of alternative hypothesis may also drop after each unsuccessful hypothesis because an even greater level of investigation may be called for.

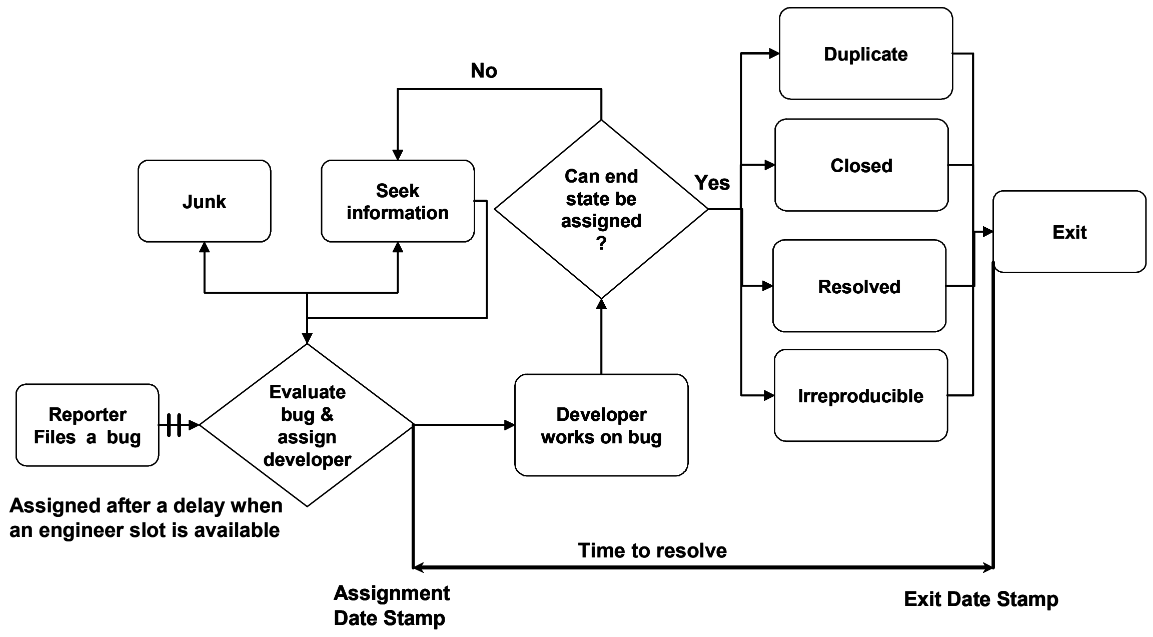

Figure 3 # Bug resolution process

Further, if the drop in the ratios of successfully resolved bugs with increasing time period as shown in Table 15 is significant, then with time, the population of bugs (tasks) in an engineer's portfolio will have a high proportion of bugs with low resolution probability. The effort invested in clearing low resolution probability bugs may come at the expense of effort that could potentially be invested in resolving incoming bugs that have a greater probability of being resolved. Under situations of high capacity utilization, theory predicts longer waiting and resolution times. In line with this, Table 16 shows that the average number of days between the submission of a bug and its assignment increases considerably from 15.22 days for severity 1 bugs to 117.43 for severity 6 bugs. The lag between submission and assignment lends credence to the idea that waiting times exist in such systems, and are higher in particular for the low severity bugs. To avoid longer waiting and resolution times, we can consider implementing a cut off policy based on threshold time that a bug spends in the system without successful resolution. Next, we model the ratios of successful resolutions over time shown in Table 15, to test whether the drop is significant using competing model specifications.

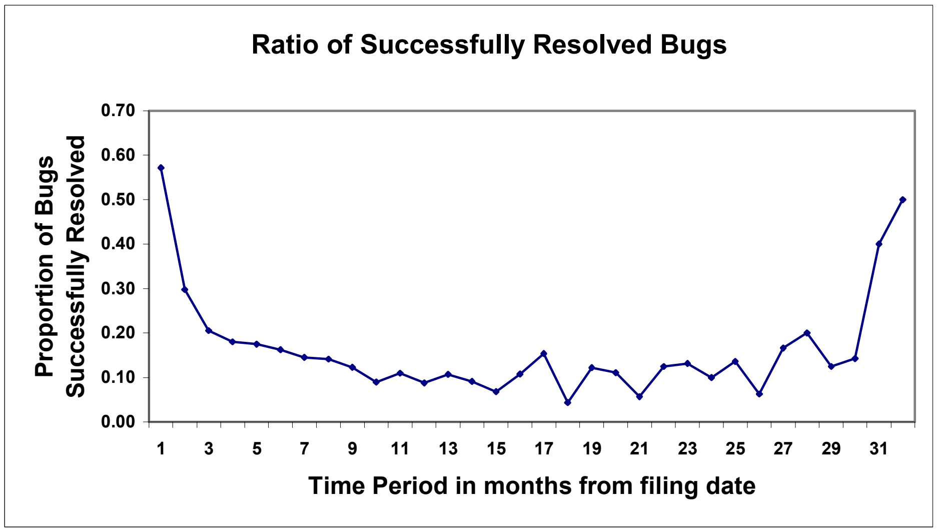

Figure 4 # Proportion of Successfully Resolved bugs in any time period

Time period (Months)	Successful Resolutions	Total	Closed/ Irreproducible	Carried Forward	Ratio Resolved
1	7147	12503	1774	5356	0.572
2	1067	3582	431	2515	0.298
3	428	2084	215	1656	0.205
4	260	1441	103	1181	0.18
5	189	1078	51	889	0.175
6	136	838	40	702	0.162
7	96	662	42	566	0.145
8	74	524	26	450	0.141
9	52	424	15	372	0.123
10	32	357	14	325	0.09
11	34	311	27	277	0.109
12	22	250	13	228	0.088
13	23	215	5	192	0.107
14	17	187	8	170	0.091
15	11	162	12	151	0.068
16	15	139	7	124	0.108
17	18	117	7	99	0.154
18	4	92	6	88	0.043
19	10	82	9	72	0.122
20	7	63	3	56	0.111
21	3	53	2	50	0.057
22	6	48	4	42	0.125
23	5	38	3	33	0.132
24	3	30	5	27	0.100
25	3	22	3	19	0.136
26	1	16	3	15	0.063
27	2	12	0	10	0.167
28	2	10	0	8	0.200
29	1	8	0	7	0.125
30	1	7	1	6	0.143
31	2	5	1	3	0.400
32	1	2	1	0	0.500
Total	9672	**25362**			0.381

Table 15 # Successful resolutions in each time period

Severity	Windsorized Mean	Median	Mean
1	8.5	2	15.22
2	12.8	3	27.85
3	20.12	3	44.39
4	46.86	7	82.16
5	59.61	9	115.20
6	62.54	3	117.43

Table 16 # Average waiting time (in days) before bug is assigned to engineers

4.3. Alternative model formulations and results

Table 15 summarizes the data for an overall 25362 bug-months that 12503 bugs spent in the system. To model the probability of successful resolution in each time period, a straightforward – naïve approach – would be to consider each entering bug having a homogeneous probability of resolution *p* in any time period[22]. In this case, the probability of resolution of a bug at any time *t* given unsuccessful resolution until *t-1* is distributed geometric. The maximum likelihood estimate of *p* for this model evaluated using data given in Table 15 is 0.381 (0.003)[23]. This number is consistent with the maximum likelihood estimate of the resolution probability provided by Table 15 (9672/25362=0.381). A plot of the expected and the actual number of resolutions is shown in Figure 5. The estimates of the log likelihood and the chi-square are shown in Table 17. The plot between the expected and actual value and the chi-square statistic do not suggest a good fit for the model.

[22] We assume that there is at least one attempt made each month the bug is in the system

[23] Number in parenthesis is standard error

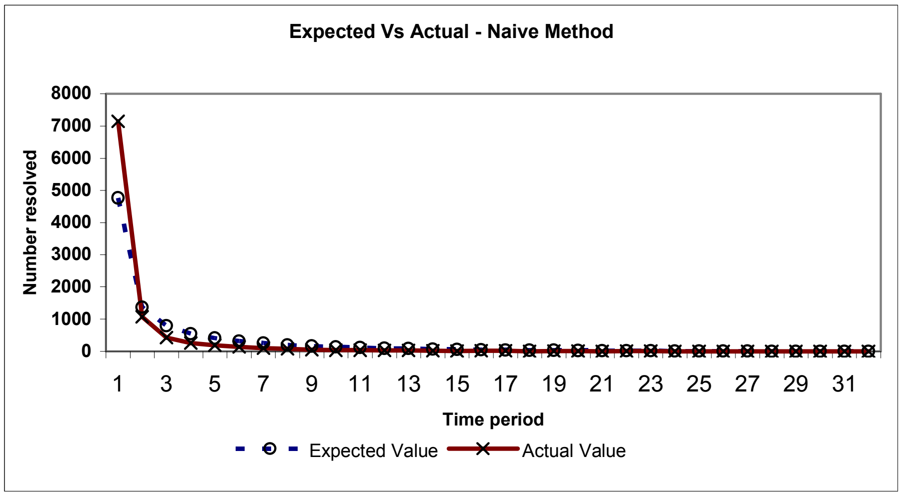

Figure 5 # Actual vs. Expected numbers for the naïve model

Further, the reduction in the ratio of bugs resolved over time suggests heterogeneity in the population of bugs, i.e. the conditional probability that a bug is resolved in time *t* given that it has not been resolved until *t-1*, is a decreasing function of time. This heterogeneity has also been observed in other contexts such as fertility studies (Kaplan et al. 1992; Weinberg and Gladen 1986; Suchindran et al. 1974). Further, the significant reduction in probability of conception success with increasing trials has operational implications for managing the flow of entities in service system such as IVF-ET centers in hospitals as shown in Kaplan et al. (1992). We draw from this stream of literature and model the presence of heterogeneity in the incoming bugs using competing model specifications and assess comparative fit of the models using multiple criterions (graphical comparison, Pearson chi-square and log likelihood values – AIC Criterion).

Beta-geometric model

A generalized expression for the probability of resolution in period *t* can be written as:

$$p_t^{'} = \int_0^1 p^t (1-p)^{t-1} f(p) dp \tag{1}$$

where, *f(p)* is the probability density function of *p*. Assuming that *p* is drawn from a beta distribution with parameters α and β, (Suchindran et al. 1974) we can show that the conditional probability of successfully resolving a bug in period *t* given that it has not been successfully resolved until period *t-1* is:

$$p_t = \frac{\alpha}{\alpha + \beta + t - 1} \tag{2}$$

In equation *2*, p_t is monotonically decreasing with time, and $p_t \rightarrow 0$ as $t \rightarrow \infty$. The parameter estimates of α and β and the standard errors of estimates for the parameters for this model are presented in Table 17 (Beta-Geometric hazard case). Figure 6 shows the relative fit of the actual and the expected values.

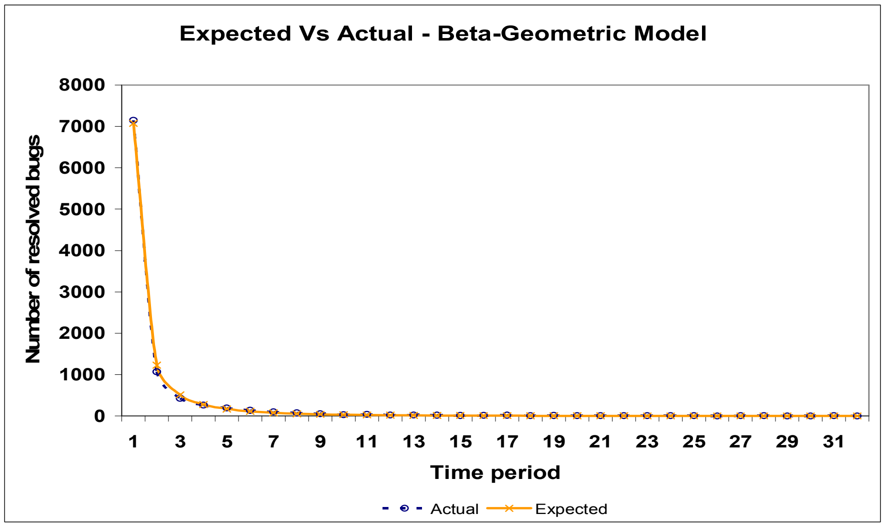

Figure 6 # Actual vs. Expected success with p distributed beta-geometric

Split population geometric model

Next we consider a split population geometric model (Kaplan et al. 1992). This model assumes that there is a fraction θ of the population of incoming bugs cannot be resolved. The probability of successful resolution in time *t* (p_t), follows geometric distribution given by $p_t = p^t(1-p)^{t-1}$. The conditional probability of successfully resolving the bug in time *t* *given* that the bug has not been resolved until *t-1* is given by

$$p_t = \frac{\theta p\,(1-p)^{t-1}}{1-\theta+\theta(1-p)^{t-1}} \tag{3}$$

Similar to equation *2*, equation *3* also has the property that $p_t \to 0$ as $t \to \infty$. The plot showing the fit between predicted and actual values of the proportion of bugs resolved in each period is shown in Figure 7. The parameter estimates of p and θ, the standard errors, and the fit measures for this model are shown in Table 17.

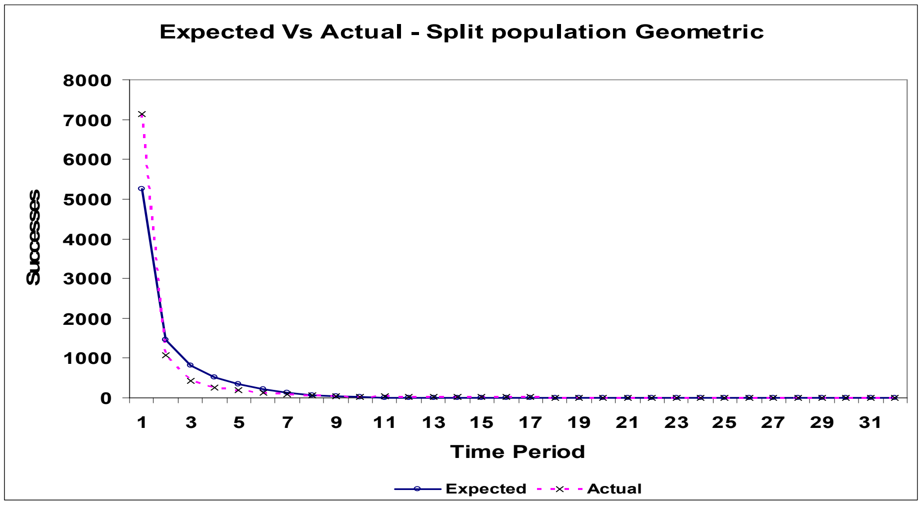

Figure 7 # Expected vs. Actual for the Split Population Geometric Model

Trinomial model

The split population model and the beta-geometric model do not explicitly parameterize the bugs that are declared unsuccessfully resolved in each time period. A variant of the split population model can be created to explicitly account for this set of bugs. In this model, in each period, there are three groups. The first group of bugs is that which is successfully resolved, the second group of bugs is that which is declared as unsuccessfully resolved and the third group of bugs are those on which no decision is reached and these bugs are carried forward into the next period. We assume that the probability of resolution p in time t is given by the beta-geometric hazard – equation 2 – and the fraction of bugs that are unsuccessfully resolved in each time period by θ. We estimate three parameters – namely α, β, and θ – to completely characterize the distribution of successfully resolved bugs in each period in this case. The likelihood function is shown in the appendix (equation 34). The model fit statistics and the parameter estimates with respective standard errors are shown in Table 17. The relative fit of the actual and expected values is shown in Figure 8.

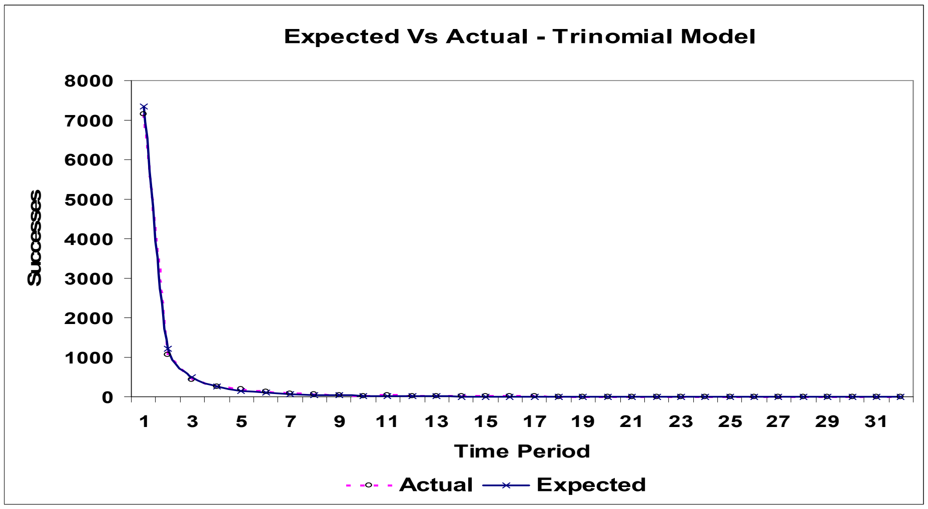

Figure 8 # Expected vs. Actual for the Trinomial Model

To compare the maximum likelihood models, we can use the Akaike Information Criterion (AIC) – $(-2LL+k)$ – where k is the number of parameters in the model, and LL is the Log-Likelihood as a measure. However as a single measure, AIC is not interpretable as a measure of model fit (Burnham and Anderson 2004). Based on this criterion, we choose the model with the lowest value of AIC – See Table 17, the valued of AIC for beta-geometric model are the least. The beta-geometric hazard model also has the lowest Chi-Square number – implying that the deviation between the actual and the expected values is least amongst the competing models. A simple but powerful evidence of the relative model fit can be to compare the plots of the predicted and actual numbers of successfully resolved bugs for the naïve, split population geometric, beta-geometric hazard and the trinomial models (Figure 5 through Figure 8). A comparison of the plot reveals that the beta-geometric distribution performs well. Further, for all the models the p-value of the likelihood ratio chi-square test for are significant ($P < 0.001$) – suggesting that the test rejects the null hypotheses that the specified model predicts the data well. However, this should not come as a surprise given that chi-square test is known to be sensitive to sample sizes[24](Segars and Grover 1993). To overcome this problem, literature suggests the use of Normed Chi-Square as a measure of model fit Segars and Grover (1993). In majority of the cases we find that the normed chi-square measure is less than the suggested metric of three. Finally, one may expect the three parameter model to outperform the two parameter model, however in our case, this expectation may not hold true because the beta-geometric and the trinomial models are non-nested. Based on the above rationale, we choose beta-geometric model for our analysis.

One of the implications of selecting the beta-geometric model is that, as $t \to \infty$, $p_t \to 0$. As stated earlier, when the teams are working at near 100% utilization, both the waiting and service times in the system will increase exponentially. In further sections, we consider the implications of imposing cut off policies on the resolution probability and the waiting times in the system.

Model	**Chi-Square**	**LL**	**Parameter**	**Estimates**	**Std. Error**
Naïve	2517.17	-16858.75	p	0.381	0.003
Beta Geometric Hazard	223.96	-14777.4	α	1.025	0.024
			β	0.543	0.023
Split Population Geometric	4,976,653	-16308.15	θ	0.963	0.003
			p	0.436	0.002
Trinomial	258.38	-21658.1	α	0.904	0.026
			β	0.426	0.018
			θ	0.11	0.001

Table 17 # Estimates and Model fit of comparative models

[24] We ran the beta-geometric model with sub-groups of data drawn from individual teams. The beta-geometric specification performed very well and the chi-square test supports the null hypothesis that the model describes the data (For example in Table 18, the data for Deployment-related category accepts that null hypothesis that the underlying distribution is beta geometric – p value is 0.621). Finally as suggested by Segars and Grover (1993) – Normed Chi-Square -- our values of Chi-square/DF for most model other than the common model exhibited the property of Chi-square/DF being less than 3 for most models with reasonable sample sizes.

4.4. Impact of cut off policies

In this section, we explore the implications of imposing a threshold time period to cut off working on a bug (denoted by *a*) and move it to an unsuccessfully resolved state. In part the methodology follows Kaplan et al. (1992). We estimate implications of this policy for the common model[25] that does not differentiate the incoming bugs in any manner. Later, in § 5, we also extend the analysis to a setting where bugs are differentiated based on the group that filed the bug.

Let U – a random variable – be the time period when a bug exits the system as unsuccessfully resolved. Let S – a random variable – is the time period when a bug exits the system as successfully resolved. Finally, let X be the random variable indicating the actual number of periods taken to resolve the bug. Then assuming U and S are independent:

$$X = \text{Min}\ \{U, S\}\ \text{and,}$$

$$\Pr\{X \geq x\} = \Pr\{U \geq x\}\Pr\{S \geq x\} \quad x = 1,2,3,.... \tag{4}$$

The number of time periods a bug is expected to last is given by

$$e = \sum_{x=1}^{\infty} \Pr\{X \geq x\} \tag{5}$$

When bugs are moved to an unsuccessfully resolved state after a threshold of *a* time periods, the probabilities $\Pr\{U(a) > a\} = 0$ and $\Pr\{S(a) > a\} = 0$, can be written as a function of the cut off time period *a*. If a bug enters period *a,* it implies that it did not reach either resolved or unresolved states until period *a-1*. The probability that a randomly chosen bug gets resolved under a threshold policy of *a* time periods is given by:

[25] Common model refers to the combined dataset of 12503 bugs

$$P(a) = \sum_{x=1}^{a} \Pr\{X(a) \geq x\} p_x \qquad (6)$$

Where, p_x is the conditional probability given by equation *2*. Assuming that the arrival process is Poisson with rate λ per month, and n engineer slots available each month for assignment of incoming bugs, and a utilization rate of $\rho(a)$, $n\,\rho(a)$ bugs will be worked on at any point in time. The exit rate of bugs from the system is given by $n\frac{\rho(a)}{e(a)}$. In steady state, the condition for stability of the queue is:

$$\rho(a) = \frac{\lambda\, e(a)}{n} < 1 \qquad (7)$$

Using equation *7*, we formulate an optimization problem to maximize the overall number of successful resolutions given an available capacity of *n* engineer slots as shown below:

$$\underset{a}{Max} \quad \lambda\, P(a) \qquad (8)$$

$$\textit{Subject to} \quad \lambda e(a) < n \quad \text{a} = 1,2,3,4\ldots \qquad (9)$$

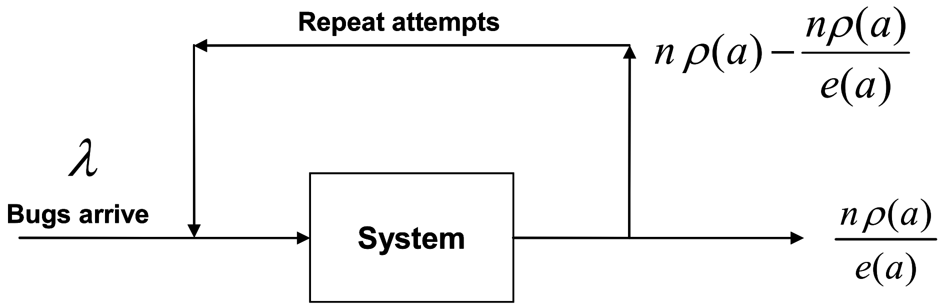
Repeat attempts
$n\,\rho(a) - \frac{n\rho(a)}{e(a)}$
λ
Bugs arrive
System
$\frac{n\,\rho(a)}{e(a)}$

Figure 9 # The queuing process

To maximize equation *8*, we need to find the *a*, that meets the constraints in equation *9*. Also, the maximum number of time periods a bug is allowed to stay in the system under a threshold policy of *a* months before declaring the bug as unsuccessfully resolved can be written as:

$$\Pr\{U(a) \geq x\} = \begin{cases} \prod_{t=1}^{x-1}(1-d_x), & a = 1,2,3...... \\ 0 & x > a \end{cases} \tag{10}$$

where, d_x is the conditional probability of moving the bugs to unsuccessfully resolved at the end of month *x* given that no decision has been reached on the state of the bug until month *x-1*. Using equation *4* and equation *10*, we can rewrite equation *6* as

$$P(a) = \sum_{a=1}^{\infty}\left(\prod_{x=1}^{a-1}(1-d_x)\Pr\{S(a) \geq x\}\left(\frac{\alpha}{\alpha+\beta+a-1}\right)\right) \tag{11}$$

where $\Pr\{S(a) \geq x\} = \prod_{j=1}^{x-1}(1-p_j)$ and $p_j = \frac{\alpha}{\alpha+\beta-j+1}$.

Given that the resolution can be cut off at $t = a$, expected time a bug stays in the system as a function of the threshold time period *a* is:

$$e(a) = \sum_{x=1}^{a}\Pr\{X(a) \geq x\} \tag{12}$$

In equation *12*, $e(1) = 1$, and $e(\infty)$ is the expected number of time periods a bug stays in the system when cut off policy is not imposed. Figure 10 and Figure 11 show the changes in $e(a)$ and $P(a)$ with increasing *a*. One can see in Figure 11, by the end of Month 7, *P(a)* is 97.3 % of its peak value of 0.77.

Our dataset of 12503 bugs spanned a total of 97 months and came from 31 different teams. Treating them as a single group, and assuming steady state arrival of bugs implies an arrival rate of, λ = 12503/97 = 128.9/Month across all the projects. The maximum likelihood estimate for d_x was computed from Table 15 based on the proportion of bugs that were moved to *unsuccessfully resolved* state in period *x* conditional on no decision until period *x-1*. The maximum likelihood values of α and β are shown in Table 17. The expected number of months taken to resolve under no threshold policy, $e(\infty)$ is 2.047. The expected number of months the bugs are in the system based on data shown in Table 15 is 25362/12503 = 2.028. Further, $P(\infty)$ – the proportion of bugs successfully resolved – as obtained from the model is 0.77. From Table 15, we can see that this turns out to be 9672/12503= 0.77. The above numbers suggest that the aggregate model describes the phenomenon well. Finally, the total number of engineer slots required is given by $\lambda e(\infty) = 128.9 * 2.204 = 263.88 \cong 264$ engineer slots per month. In the next section, we understand the implications of imposing threshold cut-off time period in a multiple group setting.

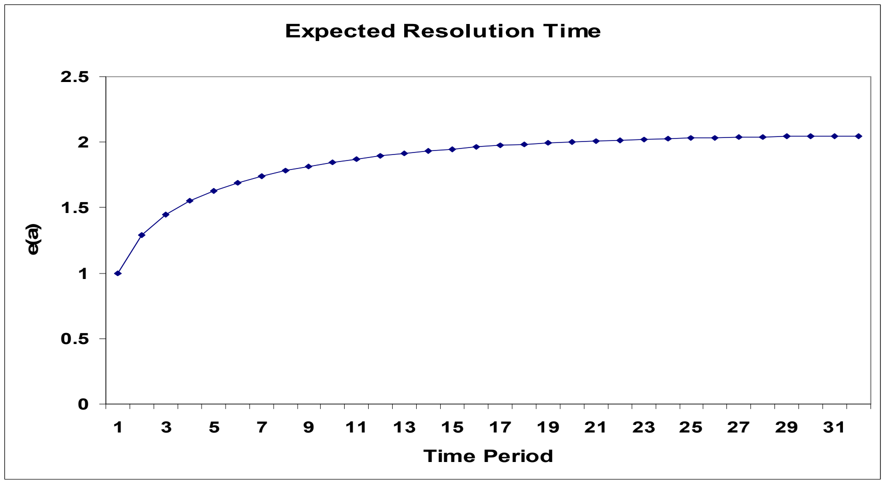

Figure 10 # Expected Resolution Time

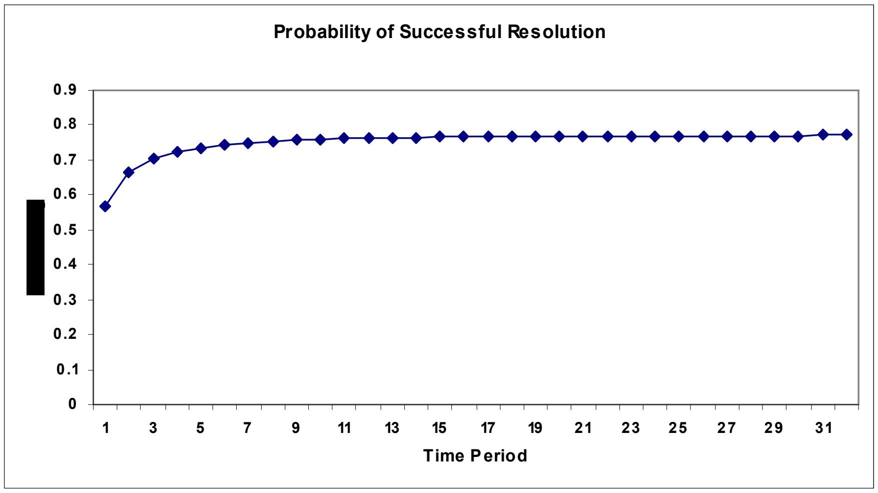

Figure 11 # Cumulative probability of successful resolution with time

4.5. Multiple group model

In this section, we relax the assumption that all the bugs come from a single group. We segregate our data based on the bug source. We identified six different groups that filed the bugs (see Table 18 for detailed breakup of the groups, the parameter estimates – α_i and β_i, standard errors, Chi-square and log likelihood for each subgroup). These sources of reported bugs differed based on the stage of the development cycle in which the bugs are filed and consequently the knowledge of the bug filer about the specifications and the code base of the software. These sources were divided into those found during development/maintenance, development-related activities (compilation, compliance testing, code inspection), actual customer deployment, deployment-related testing (alpha testing and beta testing), regression testing (involves performing integration testing on the product before it is released to customers) and unit testing or Devtest (involves testing individual software components prior to integration). Table 18 also displays the percentage of successfully resolved bugs in each of the above categories. Any rule for allocating resources must specify how available capacity is allocated across the sub-groups of bugs.

For the multi-group case equations *8* and equation *9* can be modified to:

$$\underset{a_i}{Max} \quad \sum_{i=1}^{k} \lambda_i \, P_i(a_i) \tag{13}$$

$$Subject\ to \sum_{i=1}^{k} \lambda_i e_i(a_i) < n \quad \forall\ k=1,2,3,4 \ldots \mathrm{m} \tag{14}$$

From equations *13* and *14*, for any given slot size, we can determine the maximum resolution rate by enumerating all the feasible cut-offs over the 6 groups in question. However, enumerating all possible combinations of the cut-off time periods lead to 32^6 (1,073,741,824) combinations (6 groups and a possible maximum of 32 time periods for each group). To reduce the number of enumerations, we can aggregate individual groups to form a meta-group. We now present the grouping rationale.

Creating Meta-groups

From a software engineering perspective, the six sources of bugs described earlier can be divided into three meta-groups. The first meta-group comprises bugs discovered during development and development-related activities (including compilation, compliance examination, and code inspection tasks). These activities are usually performed by development/maintenance engineers who work closely with the software code. On account their good understanding of the code, these engineers can provide relatively precise information and guidance to the maintenance engineers work on bug resolution. Therefore, the bugs filed within this meta-group can be expected to have a high fraction of successfully resolved bugs as compared to those filed by other entities. The second meta-group comprises bugs discovered during regression and unit testing. These types of testing are performed when various software components have been integrated, or when a completed module needs to be tested for its functionality. Bugs discovered during such testing can span multiple components, or multiple code bases within a component. Therefore, these bugs can be more difficult to reproduce and resolve. Finally, the third meta-group comprises bugs filed by customers and bugs which are deployment-related (the latter could include bugs discovered during the alpha and beta testing stages). Customers are least familiar with the software code. Further, during the alpha and beta testing stages and during the live deployment of the software at customer locations, the software may be put to work in diverse environments not encountered during the formal testing process. This may lead to the discovery of bugs which are either not well specified or not easily reproducible in the lab environment. Thus, these bugs are least likely to be successfully resolved.

We now check this three-way grouping empirically. Consider the percentage of successfully resolved bugs across the meta-groups in Table 19. As expected, development and development-related bugs have the highest percentage of successfully resolved bugs (See actual fraction of bugs resolved in each subgroup), Customer found and deployment-related bugs have the least fraction of bugs resolved and finally the testing bugs fall between the other two. Further, we can empirically demonstrate the equivalence of the pairs of sources which are combined within each meta-group by comparing parameter estimates (α and β) across the source-pairs. To verify this grouping rigorously, we investigated the equivalence of the parameters α and β by considering a pair of values in any given time and tested whether they were significantly different from each other[26]. If both α and β are *not* significantly different from each other for individual groups, then the groups are equivalent. In that case, we created a new subgroup by treating the two groups to be the same and re-estimated α and β for the combined group. If either α or β for two groups *are* significantly different from each other, we treat them as distinct groups.

Our tests of parameter equivalence yielded three distinct meta-groups. Interestingly, these groups are consistent with earlier theoretical observations of grouping by stage of the development life cycle. The estimates of α and β, the log-likelihood, and the Pearson's chi-square for each of the meta-group is presented in Table 19. In addition to the calculated estimates of α and β, Table 19 also presents the data for the predicted and actual times the bug stays in the system, and the predicted and actual probability of a bug being successfully resolved, for each meta-group. The closeness of the predicted and actual estimates in Table 19 suggests validation of the beta-geometric approach in individual meta-groups.

---

[26] This was done by a simple transformation of the beta geometric distribution to $\frac{1}{\mathrm{p_t}} = \mathrm{a} + \mathrm{b(t-1)}$ based on Weinberg and Gladen (1986). We can introduce a covariate Z and write the function as $\frac{1}{p_t} = a_1 + b_1(t-1) + Z(c + d(t-1))$. This enables us to test for the significance of coefficients (obtained using maximum likelihood) c and d to estimate the equivalence between the groups.

Main Group	Sub Group	Sample Size	α	β	Chi-Square(DF)	Log-Likelihood	% Successfully Resolved
Common Group	**Common Group**	12503	1.025	0.543	223.96 (31)	-14777.4	77.30%
	Standard Error		0.024	0.023			
Grouping by the Filer of the Bug	**Development**	2245	1.031	0.543	73.58 (31)	-2751.54	91.63%
	Standard Error		0.066	0.046			
	Regression and Automation	1818	0.836	0.616	40.23 (22)	-1860.52	72.28%
	Standard Error		0.084	0.073			
	Customer Found	2119	0.871	1.198	75.69 (31)	-2964.77	69.80%
	Standard Error		0.067	0.12			
	Devtest	5023	0.795	0.575	115.71 (31)	-5445.33	74.74%
	Standard Error		0.043	0.038			
	Development-related	729	1.207	0.663	40.80 (20)	-864.035	87.40%
	Standard Error		0.154	0.107			
	Deployment-related	512	0.97	1.087	27.99 (31)	-698.409	74.41%
	Standard Error		0.369	0.145			

Table 18 # Maximum Likelihood estimates of α *and* β

Group	Item	Estimate	Log Likelihood	Chi-Square
Development and Development-related	Arrival Rate	27.12/Month		
	α (S.E)	1.062 (0.060)		
	β (S.E)	0.565 (0.042)		
	Predicted expected time to successful resolution	2.08 Months	-3636.52	101.68 (31)
	Actual expected time to successful resolution	2.07 Months		
	Predicted probability of successful resolution	0.9		
	Actual fraction successfully resolved	0.905		
Regression and Devtest	Arrival Rate	70.52/Month		
	α (S.E)	0.804 (0.038)		
	β (S.E)	0.585 (0.033)		
	Predicted expected time to successful resolution	1.805 Months	-7305.96	143.88 (31)
	Actual expected time to successful resolution	1.783 Months		
	Predicted probability of successful resolution	0.738		
	Actual fraction successfully resolved	0.74		
Customer Found and Deployment-related	Arrival Rate	31.24/Month		
	α (S.E)	0.887 (0.061)		
	β (S.E)	1.172 (0.105)		
	Predicted expected time to successful resolution	2.62 Months	-3665.81	81.30 (31)
	Actual expected time to successful resolution	2.61 Months		
	Predicted probability of successful resolution	0.703		
	Actual fraction successfully resolved	0.706		

Table 19 # Maximum Likelihood estimates and related data for sub-groups

Enumeration of cut-offs

The creation of meta-groups simplifies the number of enumerations required considerably. As against the 32^6 (1,073,741,824) enumerations required earlier, we now have to consider only 32^3 (32,768) enumerations to choose the optimal cut-off combination. We now present the algorithm for choosing the optimal cut off. This algorithm is based on Kaplan et al. (1992).

```
Initialize: Set LOWERBOUND RR = 0, $a_i^*(n)=0$, $a_j^*(n)=0$, $a_k^*(n)=0 \ \forall\ i, j, k=1, 2, 3$
    Set UPPERBOUND $a_i^{\max}(n)$, $a_j^{\max}(n)$, $a_k^{\max}(n) \ \forall\ i, j, k=1, 2, 3$
    OUTER LOOP: FOR $i = 1,2,3$ $a_i(n) = 1, 2, 3, 4, \ldots a_i^{\max}(n)$
        MIDDLE LOOP: FOR $j = 1,2,3$ and $j \neq i$, $a_j(n) = 1,2,3,4\ldots.. a_j^{\max}(n)$
            INNER LOOP: FOR $k = 1,2,3$, $k \neq i$, $k \neq j$ $a_k^{\max}(n) = e_k^{-1}\left(\frac{(n - \lambda_i e_i(a_i(n)) - \lambda_j e_j(a_j(n)))}{\lambda_k}\right)$
                If $\lambda_i e_i(a_i(n)) + \lambda_j e_j(a_j(n)) + \lambda_k e_k(a_k(n)) < n$
                    if $\lambda_i P_i(a_i) + \lambda_j P_j(a_j) + \lambda_k P_k(a_k) > RR$ then
                        set $a_i^*(n) = a_i(n)$, $a_j^*(n) = a_j(n)$, $a_k^*(n) = a_k(n)$
                        Set $\lambda_i P_i(a_i) + \lambda_j P_j(a_j) + \lambda_k P_k(a_k) = RR$
```

The optimal combination of $a_i(n)$ for a given *n* (slot size) is one that maximizes the overall resolution rates for a given slot size. Figure 12 shows the optimal cut-off schedule for individual groups with increasing slot size. In particular, when the number of slots is lower, it reflects the tradeoffs that need to be made in imposing the cut off months on a type of bug. From Figure 12, we see that when the available number of slots is low, the resolution rate can be maximized by focusing on developer filed bugs. Intuitively, this is consistent with the observation in our data that the probability of *successful resolution* of developer filed bugs is high (0.90). Note that in this allocation, all the incoming bugs from any source in a time period are treated equally, i.e., if even one bug from that source is admitted then all the bugs from that source should be admitted. For example, in period 1 if the management has 15 slots and the minimum number of slots needed to accommodate all the incoming arrivals of the developer filed bugs in period 1 is 28 slots, none of the bugs will be accommodated. We label this approach as the "*equality rule*." However, this approach need not necessarily yield us the maximum resolution at any point in time.

To maximize resolution rates for any given capacity without imposing the equality rule, one can follow a scheme where bugs are allocated based on the highest marginal resolution probability in each time period. We call this allocation scheme the "*marginal probability rule*." In this case, given the availability of 30 slots, after completing the allocation of the slots to the bugs with highest probability of resolution – For example, developer filed bugs in the first time period because of highest resolution probability in the period –, any left over slots are given to the bugs with the next highest marginal resolution probability – In this case, tester filed bugs in the first time period. However, note that, not all arrivals of the second group can necessarily be served and to the extent that the slots for assignment are available a fraction of the bugs is served. For time periods that are greater than one month, we can determine the number of bugs in steady state that are carried forward to the next period (for example period 2) and assign only a fraction of the bugs for which the slots are available in that period. Table 20 shows a comparison of the resolution rates between the "marginal probability rule" and the "equality rule". The following example illustrates the interpretation of the results shown in Table 20. For 150 slots, the optimal policy would impose a cut off period of 2 months on development and development-related bugs, 2 months on regression and unit test bugs and one month on customer and deployment-related bugs. Further, the policy indicates that engineers should work on *all* development and development-related bugs until the end of second period, work on *all* regression and unit test bugs until the end of first period *and* carry forward only 76% of the leftover bugs from this meta-group to be worked on in the second period, and finally, work on all customer and deployment-related bugs till the end of the first period (no bugs in this last meta-group are carried forward into the second period). Finally, note that as expected, the "marginal probability rule" allocation performs better in terms of the resolution rates particularly when the slot sizes are low. However, for all slot size numbers greater than 150, both the rules perform equally well. From a managerial perspective, however, implementing the "marginal probability rule" requires keeping track of the number of bugs that one needs to work on in a particular category for any slot size.

Table 20 also shows the cumulative resolution rates based on the "*equality rule*." Figure 13 shows the cut off combinations for the maximum resolution rate for the "*Marginal Resolution rule*" case. Figure 14 shows the relative percentage of bugs resolved – computed as $\left(\frac{\lambda_1 P_1(a_1)+\lambda_2 P_2(a_2)+\lambda_3 P_3(a_3)}{\lambda_1 P_1(\infty)+\lambda_2 P_2(\infty)+\lambda_3 P_3(\infty)}*100\right)$ – with increase in number of slots available. This shows that the marginal rate of resolution of bugs reduces as the slot sizes increase indicating that the sensitivity of addition of capacity on overall rates of successful resolution is low as more people are added into the teams.

Maximum Integer Cut offs				Fraction of Bugs considered for resolution in the "Maximum Integer Cut off period"			Resolution Rate -- Marginal Probability rule	Resolution Rate -- Equality rule
Slots	Devel-opment	Testing	Customer	Devel-opment	Testing	Customer		
30	1.00	1.00	0.00	1.00	0.04	0.00	19.37	17.71
35	1.00	1.00	0.00	1.00	0.11	0.00	22.27	20.88
40	1.00	1.00	0.00	1.00	0.18	0.00	25.16	22.17
45	1.00	1.00	0.00	1.00	0.25	0.00	28.06	23.25
50	1.00	1.00	0.00	1.00	0.32	0.00	30.95	23.96
55	1.00	1.00	0.00	1.00	0.40	0.00	33.85	24.37
60	1.00	1.00	0.00	1.00	0.47	0.00	36.74	31.17
65	1.00	1.00	0.00	1.00	0.54	0.00	39.64	31.17
70	1.00	1.00	0.00	1.00	0.61	0.00	42.53	34.35
75	1.00	1.00	0.00	1.00	0.68	0.00	45.43	40.83
80	1.00	1.00	0.00	1.00	0.75	0.00	48.32	40.83
85	1.00	1.00	0.00	1.00	0.82	0.00	51.22	40.83
90	1.00	1.00	0.00	1.00	0.89	0.00	54.11	46.62
95	1.00	1.00	0.00	1.00	0.96	0.00	57.01	46.62
100	1.00	1.00	1.00	1.00	1.00	0.08	59.55	58.54
105	1.00	1.00	1.00	1.00	1.00	0.23	61.71	58.54
110	1.00	1.00	1.00	1.00	1.00	0.39	63.86	63.00
115	1.00	1.00	1.00	1.00	1.00	0.55	66.02	64.33
120	1.00	1.00	1.00	1.00	1.00	0.71	68.17	64.69
125	1.00	1.00	1.00	1.00	1.00	0.87	70.32	67.50
130	2.00	1.00	1.00	0.12	1.00	1.00	72.45	72.00
135	2.00	1.00	1.00	0.35	1.00	1.00	72.45	72.00
140	2.00	2.00	1.00	1.00	0.18	1.00	76.29	75.18
145	2.00	2.00	1.00	1.00	0.47	1.00	77.97	77.13
150	2.00	2.00	1.00	1.00	0.76	1.00	79.66	78.93
155	3.00	2.00	1.00	0.21	1.00	1.00	81.30	80.97
160	3.00	2.00	2.00	1.00	1.00	0.12	82.76	82.25
165	3.00	2.00	2.00	1.00	1.00	0.50	84.21	83.33
170	3.00	2.00	2.00	1.00	1.00	0.89	85.66	84.71
175	3.00	3.00	2.00	1.00	0.39	1.00	86.93	86.67
180	3.00	3.00	2.00	1.00	0.95	1.00	88.11	87.64
185	4.00	3.00	3.00	1.00	1.00	0.13	89.25	88.79
190	4.00	3.00	3.00	1.00	1.00	0.88	90.34	89.85
195	5.00	4.00	3.00	1.00	0.34	1.00	91.30	91.12
200	5.00	4.00	4.00	1.00	1.00	0.16	92.21	92.00
205	5.00	4.00	4.00	1.00	1.00	0.82	93.07	92.80
210	6.00	5.00	5.00	1.00	1.00	0.05	93.82	93.69
215	7.00	5.01	5.00	1.00	0.01	1.00	94.54	94.43
220	7.00	6.00	6.00	1.00	1.00	0.57	95.17	95.05
225	9.00	6.00	6.00	1.00	1.00	0.64	95.76	95.65
230	10.00	7.00	8.00	1.00	1.00	0.28	96.29	96.19
235	11.00	8.00	9.00	1.00	1.00	0.34	96.76	96.65
240	12.00	9.00	10.00	1.00	1.00	0.90	97.19	97.08
245	14.00	11.00	11.00	1.00	0.63	1.00	97.56	97.47
250	16.00	12.00	13.00	1.00	1.00	1.00	97.89	97.79
255	19.00	15.00	16.00	1.00	1.00	0.36	98.17	98.07
260	24.00	18.00	20.00	1.00	1.00	0.27	98.41	98.31

Table 20 # Integer Cutoffs and Fraction considered for resolution based on Marginal Probability Rule

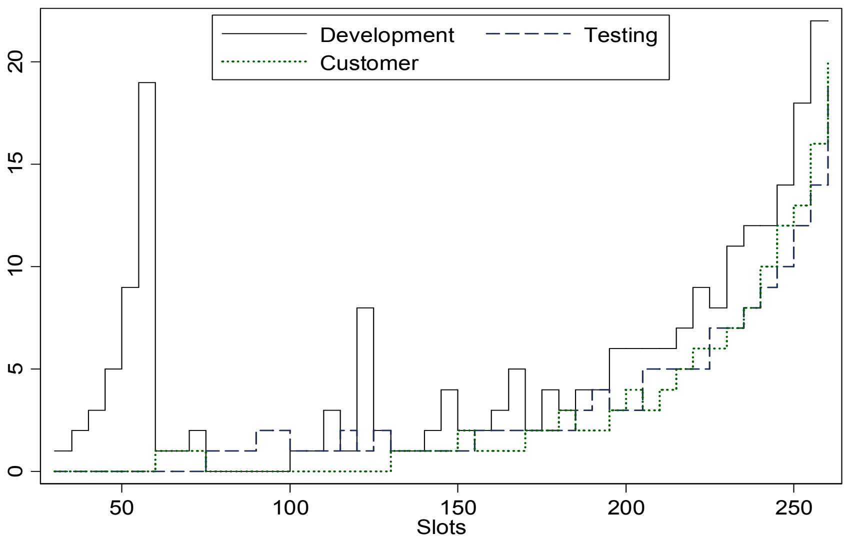

Figure 12 # Optimal cut off combinations for different groups given slot size – Equality Rule

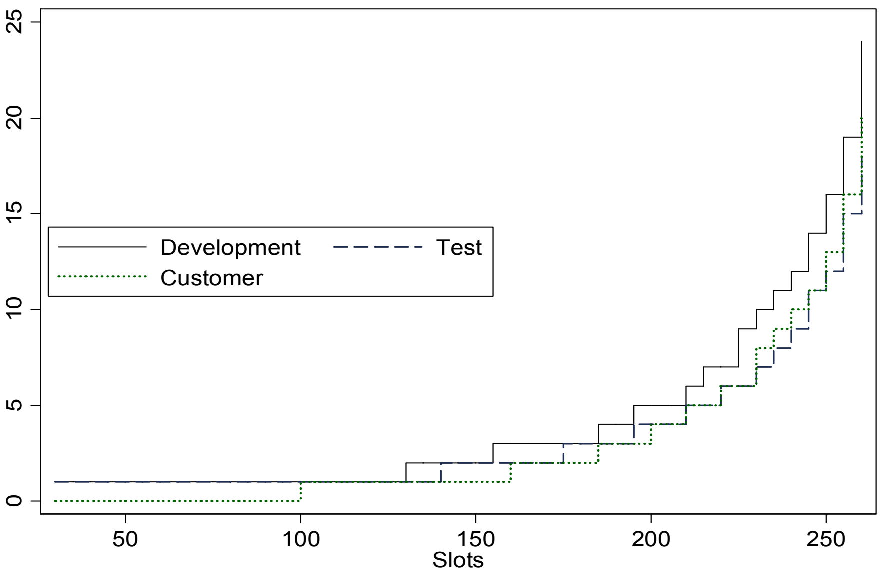

Figure 13 # Optimal cut off combinations for different groups given slot size – Marginal Probability Rule

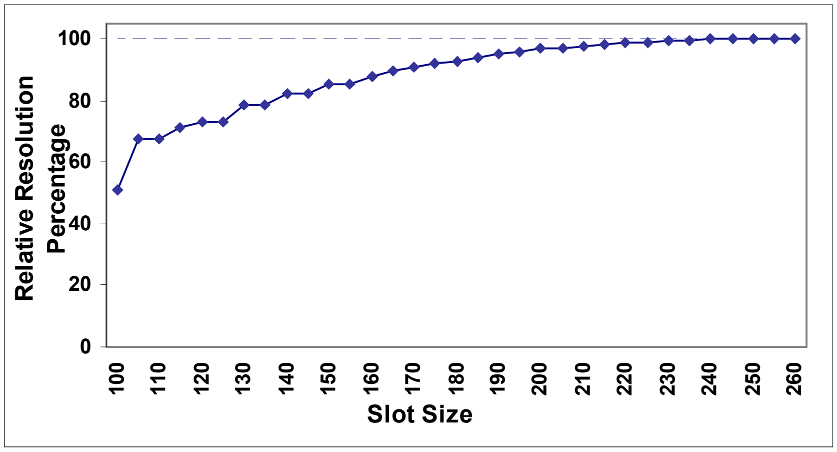

Figure 14 # Relative resolution rates for the three groups based on slot size

4.6. Waiting Time tradeoffs

In this section, we analyze the effects of imposing the cut off policies on the waiting times of bugs in the system. Longer queues not only lead to assignment delays, but also lead to resolution delays. Using approximations developed in prior literature, we analyze two different cases. First, we analyze the case where bugs are handled on a First Come First Serve (FCFS) basis. Next, we analyze the case assuming the incoming population follows a preemptive resume priority queue with higher severity bugs preempting the lower severity bugs.

Non-preemptive FCFS case

Let $W_{NP}(a)$ be the total time a bug spends in the system when a threshold policy of a time periods is enforced in the non-preemptive case. Therefore:

$$W_{NP}(a) = e(a) + W_Q(a) \tag{15}$$

$W_Q(a)$ is the waiting time in the queue with a threshold policy of *a* time periods and *T(a)* is the expected time that the bug spends in service – as calculated earlier. We assume that the arrival process of the bugs is poisson – with rate λ – consistent with related literature in software reliability (van Pul 1994). We assume the queue to be an M/G/n queue.

The waiting times in queue for the non-preemptive FCFS case can be computed for the M/G/n queue using the approximation suggested by Whitt (1983). $W_Q(a)$ can be approximated by[27]

$$W_Q(a) = \frac{1}{2}\left(1 + \frac{Var(X(a))}{e(a)^2}\right)Q(\lambda, P(a), n) \tag{16}$$

Here, $Q(\lambda, P(a), n)$ is the exact delay for a M/M/n system with arrival rate λ. Table 21 shows the mean time a bug spends in the system and Table 22 gives the sensitivity analysis as the number of slots is increased from 264 to 275. One can see that the queue length reduces to zero as the slot size increases from 264 to 275. Note that the computation has an underlying assumption that each incoming bug can be assigned to any individual. However, in reality, teams are divided into individual "projects" – supervised by a lead – that may have reduced number of servers (engineers). This has a direct impact in increasing the waiting line of the bugs. To analyze the impact of this phenomenon, we show the similar analysis by using only one of the teams. The results of the waiting line and the impact of increase in number of slots in the system is shown in Table 23. This is considerably larger as compared to the waiting times shown in Table 21 implying that at a team level, such policies may have greater impact.

Preemptive resume priority case

In this section, we investigate the case where higher severity bugs can be pre-empted by bugs of lower severity – consistent with the observation that higher severity bugs have to be resolved with a greater urgency. More specifically, we grouped the bugs into three categories (*Group 1*: Severity 1 and Severity 2, *Group 2*: Severity 3 and Severity 4, and *Group 3*: Severity 5 and Severity 6). To calculate the waiting times in queue under pre-emptive resume priority, we use the method suggested by Bondi and Buzen (1984) as shown in equation *17.*[28]

$$W(PRI,\underline{\mu}_{(p)},\underline{\lambda}_{(p)},n) \approx W(PRI,n\underline{\mu}_{(p)},\underline{\lambda}_{(p)},1)\frac{W(FCFS,\underline{\mu}_{(p)},\underline{\lambda}_{(p)},n)}{W(FCFS,n\underline{\mu}_{(p)},\underline{\lambda}_{(p)},1)} \tag{17}$$

Here, $W(PRI,\underline{\mu}_{(p)},\underline{\lambda}_{(p)},n)$ is the computed waiting time for the system with *n* servers under a pre-emptive resume priority (PRI) discipline for the *p* priority classes[29], with an arrival rate vector $\underline{\lambda}_{(p)} = (\lambda_1,\lambda_2,...\lambda_p)$ and a service rate vector denoted by $\underline{\mu}_{(p)} = (\mu_1,\mu_2,...\mu_p)$.
$W(FCFS,\underline{\mu}_{(p)},\underline{\lambda}_{(p)},n)$ is the waiting time for the M/G/n system with a first come first serve (FCFS) priority. This can be computed using the approximations in Nozaki and Ross (1978), or Boxma et al. (1978).

[27] We can also compute this using other approximations by Boxma et al. (1979) and by Nozaki and Ross (1978). We verified our waiting times with all the three approximations. They yield values that are very close to each other. The expressions for the approximations are given in the appendix.
[28] Notations used in the equations are consistent with Bondi and Buzen (1984)

[29] Note that our assumption of pre-emptive resume priority queue is reasonable in this scenario given that engineers have some prior knowledge of the resolution history of the preempted bug

$W(FCFS, n\underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, 1)$ is the waiting time for the M/G/1 system with a first come first serve priority with an average service rate of $n\overline{\mu}_{(p)}$, where $n\overline{\mu}_{(p)}$ is the mean service rate weighted by the *p* priority levels is given by $\overline{\mu}_{(p)} \left(\overline{\mu}_{(p)} = \left(\sum_{j=1}^{p} \frac{\lambda_j}{\mu_j} \right)^{-1} \sum_{j=1}^{p} \lambda_j \right)$. This can be obtained using *Pollaczek-Khintchine Formula.*

$$W(FCFS, n\underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, 1) = \frac{\lambda_{(p)} E[X(a)^2]}{2n^2(1-\rho_{(p)})} \tag{18}$$

Here $\lambda_{(p)} = \sum_{j=1}^{p} \lambda_j$ for the *p* priority classes and $\rho_{(p)} = \sum_{j=1}^{p} \rho_j$. The condition of stability of the queue implies that $\rho(r) = \sum \lambda_i / n\mu_i < 1$.

$W(PRI, n\underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, 1)$ indicates the average waiting time for an M/G/1 pre-emptive priority queue with *p* priority classes and is computed as

$$W(PRI, n\underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, 1) = \frac{1}{\lambda_{(p)}} \sum_{i=1}^{p} \lambda_i \overline{W_i} \tag{19}$$

Here, $\overline{W_i}$ is the expected waiting time for individual priority class *i* and

$$\overline{W_i} = \frac{\sum_{j=1}^{i} \frac{\lambda_j E[X_j(a)^2]}{2}}{\left(1 - \sum_{j=1}^{i} \rho_j\right)\left(1 - \sum_{j=1}^{i-1} \rho_j\right)} \tag{20}$$

Finally, the approximation given in equation *17* as suggested by Bondi and Buzen (1984) is better when the queue of the high priority customers is not very long. In our data the grouping of Severity 1 and Severity 2 bugs constituted 3253/12503 = 26% of the overall number of bugs.

Bondi and Buzen (1988) find that the approximation suggested by Boxma et al. (1978) gives more accurate predictions as opposed to the approximations suggested by Nozaki and Ross (1978) when they compared the approximations to the simulated values. Accordingly we computed the waiting times based on Boxma et al. (1978). Finally, the total time in the system *W(a)* is given by:

$$W_{PP}(a) = W(PRI, \underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, n) + 1/\overline{\mu}_{(p)} \tag{21}$$

Here, $W_{PP}(a)$ is the overall waiting time for the PRI case. Table 21 provides estimates of $W_{PP}(a)$ computed based on waiting time approximation suggested by Boxma et al. (1978). We find that preemption increases the overall delay significantly. In particular this is true for the lower priority classes. This illustrates that in reality, the waiting times may be significantly large due to the nature of the process, thus reinforcing the need for cutting off working on the bugs at any particular point in time. In particular, we also find that, while the resolution times of high severity bugs are not compromised in the overall system, preemption has a serious impact on the system waiting times of the low severity bugs with increasing time periods the bugs stay in the system. For clarity, Table 24 and Table 25 enumerate the steps used to compute W_{NP} and W_{PP} shown in Table 21.

Cut off time period (a)	**Resolution Probability**	W_{NP} **(264 slots)**	W_{NP} **(265 slots)**	W_{PP} **(264 slots)**[30]	W_{PP} **(265 slots)**
1	0.566	1.00	1.00	1.000	1.000
2	0.665	1.29	1.29	1.290	1.291
3	0.704	1.45	1.45	1.449	1.449
4	0.724	1.55	1.55	1.553	1.554
5	0.736	1.63	1.63	1.630	1.632
6	0.744	1.69	1.69	1.698	1.698
7	0.750	1.74	1.74	1.781	1.772
8	0.754	1.78	1.78	1.919	1.881
9	0.757	1.82	1.82	2.178	2.072
10	0.760	1.85	1.85	2.654	2.407
11	0.762	1.88	1.88	3.478	2.968
12	0.763	1.91	1.91	4.728	3.793
13	0.764	1.94	1.94	6.592	4.985
14	0.765	1.97	1.97	9.378	6.705
15	0.766	2.01	2.00	13.387	9.079
16	0.767	2.05	2.03	18.847	12.169
17	0.768	2.09	2.07	26.349	16.193
18	0.768	2.15	2.12	36.448	21.302
19	0.769	2.22	2.18	50.031	27.702
20	0.769	2.30	2.24	67.009	35.073
21	0.769	2.41	2.31	89.900	44.073
22	0.769	2.54	2.41	121.944	55.180
23	0.770	2.72	2.52	165.156	67.926
24	0.770	2.97	2.66	225.016	82.206
25	0.770	3.26	2.81	299.609	95.844
26	0.770	3.63	2.97	394.845	108.514
27	0.770	4.08	3.14	509.075	118.875
28	0.770	4.76	3.36	686.400	128.165
29	0.770	5.95	3.64	996.681	133.695
30	0.770	8.52	4.02	1672.947	128.995
31	0.770	15.12	4.47	3402.699	119.444
32	0.770	37.66	4.87		

Table 21 # Tradeoff between mean time a bug spends in the system and the probability of a successful resolution

[30] A breakup of waiting times by severity class is shown in Table 23

Non Preemptive setting					Preemptive setting			
				Slots				
Cut off Month	264	265	270	275	264	265	270	275
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	1.29	1.29	1.29	1.29	1.29	1.29	1.29	1.29
3	1.45	1.45	1.45	1.45	1.45	1.45	1.45	1.45
4	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55
5	1.63	1.63	1.63	1.63	1.63	1.63	1.63	1.63
6	1.69	1.69	1.69	1.69	1.70	1.70	1.69	1.69
7	1.74	1.74	1.74	1.74	1.78	1.77	1.75	1.74
8	1.78	1.78	1.78	1.78	1.91	1.88	1.80	1.79
9	1.82	1.82	1.82	1.82	2.14	2.08	1.85	1.83
10	1.85	1.85	1.85	1.85	2.55	2.43	1.91	1.88
11	1.88	1.88	1.87	1.87	3.24	3.02	1.99	1.93
12	1.91	1.91	1.90	1.89	4.28	3.90	2.07	1.98
13	1.94	1.94	1.92	1.91	5.81	5.17	2.17	2.04
14	1.97	1.97	1.93	1.93	8.06	7.04	2.27	2.12
15	2.01	2.00	1.95	1.95	11.26	9.66	2.39	2.19
16	2.05	2.03	1.96	1.96	15.55	13.12	2.51	2.28
17	2.09	2.07	1.98	1.98	21.38	17.73	2.62	2.36
18	2.15	2.12	1.99	1.99	29.16	23.73	2.74	2.44
19	2.22	2.18	2.00	2.00	39.53	31.50	2.85	2.53
20	2.30	2.24	2.01	2.01	52.41	40.79	2.95	2.60
21	2.41	2.31	2.02	2.01	69.64	52.69	3.05	2.68
22	2.54	2.41	2.02	2.02	93.60	68.33	3.14	2.75
23	2.72	2.52	2.03	2.03	125.70	87.80	3.23	2.82
24	2.97	2.66	2.04	2.03	169.92	112.22	3.31	2.88
25	3.26	2.81	2.04	2.04	224.69	139.22	3.38	2.93
26	3.63	2.97	2.04	2.04	294.42	169.34	3.43	2.97
27	4.08	3.14	2.05	2.04	377.95	200.38	3.48	3.01
28	4.76	3.36	2.05	2.05	507.44	240.19	3.52	3.04
29	5.95	3.64	2.05	2.05	733.76	292.85	3.57	3.08
30	8.52	4.02	2.05	2.05	1226.61	365.47	3.61	3.11
31	15.12	4.47	2.06	2.05	2485.66	449.36	3.64	3.13

Table 22 # Sensitivity of waiting line to slot size increments

Cut off time period (a)	**Probability of resolution**	**Mean Time (52 slots) W_{NP}**	**Mean Time (53 slots) W_{NP}**
1	0.531	1.00	1.00
2	0.623	1.27	1.27
3	0.657	1.41	1.41
4	0.673	1.51	1.50
5	0.682	1.59	1.58
6	0.688	1.67	1.64
7	0.693	1.76	1.71
8	0.696	1.86	1.79
9	0.698	1.99	1.87
10	0.699	2.16	1.96
11	0.701	2.37	2.07
12	0.702	2.66	2.19
13	0.702	3.02	2.32
14	0.703	3.57	2.48
15	0.704	4.56	2.68
16	0.704	6.73	2.95
17	0.704	12.74	3.25

Table 23 # Tradeoff between mean time a bug spends in the system and the probability of a successful resolution – A single Project Case

Cut off period (a)	$\overline{W_{12}}$	$\overline{W_{34}}$	$\overline{W_{56}}$	$W(PRI, n\underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, 1)$	$W(FCFS, \underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, n)$	$W(FCFS, n\underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, 1)$	$W(PRI, \underline{\mu}_{(p)}, \underline{\lambda}_{(p)}, n)$
1	0.000	0.000	0.000	0.000	0.000	0.002	0.000
2	0.124	1.134	2.874	1.018	0.000	0.005	0.000
3	0.166	1.921	6.132	1.819	0.000	0.008	0.000
4	0.203	2.783	10.745	2.783	0.000	0.013	0.000
5	0.238	3.725	17.017	3.939	0.000	0.018	0.001
6	0.271	4.754	25.435	5.332	0.000	0.025	0.007
7	0.304	5.866	36.461	7.000	0.000	0.033	0.037
8	0.336	7.015	50.277	8.926	0.001	0.043	0.123
9	0.367	8.205	67.575	11.174	0.002	0.054	0.318
10	0.397	9.459	89.515	13.855	0.003	0.067	0.699
11	0.427	10.768	117.218	17.059	0.007	0.083	1.370
12	0.456	12.061	150.058	20.685	0.012	0.101	2.387
13	0.485	13.395	190.516	24.981	0.019	0.123	3.896
14	0.513	14.775	241.780	30.219	0.030	0.149	6.130
15	0.540	16.159	305.764	36.533	0.046	0.181	9.307
16	0.566	17.515	383.596	43.998	0.068	0.219	13.589
17	0.591	18.844	480.436	53.048	0.097	0.266	19.403
18	0.616	20.173	601.213	64.119	0.137	0.323	27.173
19	0.641	21.473	753.451	77.826	0.189	0.393	37.535
20	0.662	22.663	933.795	93.833	0.255	0.475	50.401
21	0.680	23.809	1165.350	114.132	0.343	0.579	67.625
22	0.698	24.934	1476.345	141.121	0.466	0.718	91.581
23	0.716	25.953	1877.999	175.693	0.632	0.897	123.679
24	0.730	26.889	2422.525	222.269	0.860	1.139	167.895
25	0.739	27.620	3082.253	278.432	1.144	1.431	222.653
26	0.748	28.240	3902.323	348.055	1.507	1.794	292.383
27	0.758	28.745	4858.342	429.078	1.941	2.216	375.914
28	0.767	29.253	6304.280	551.451	2.616	2.854	505.402
29	0.777	29.762	8742.947	757.613	3.794	3.929	731.721
30	0.786	30.274	13722.794	1178.255	6.363	6.122	1224.560
31	0.792	30.555	20931.922	1786.899	12.955	9.321	2483.611

Table 24 # Computation of waiting times of individual classes when preemptive resume priority is followed (264 Slots)

Cut off Period (a)	$\overline{W_{12}}$	$\overline{W_{34}}$	$\overline{W_{56}}$	$W(PRI,n\underline{\mu}_{(p)},\underline{\lambda}_{(p)},1)$	$W(FCFS,\underline{\mu}_{(p)},\underline{\lambda}_{(p)},n)$	$W(FCFS,n\underline{\mu}_{(p)},\underline{\lambda}_{(p)},1)$	$W(PRI,\underline{\mu}_{(p)},\underline{\lambda}_{(p)},n)$
1	0.000	0.000	0.000	0.000	0.000	0.002	0.000
2	0.123	1.123	2.830	1.007	0.000	0.005	0.000
3	0.165	1.900	6.014	1.795	0.000	0.008	0.000
4	0.202	2.748	10.494	2.739	0.000	0.013	0.000
5	0.237	3.673	16.552	3.865	0.000	0.018	0.001
6	0.270	4.683	24.633	5.217	0.002	0.025	0.006
7	0.303	5.771	35.147	6.826	0.008	0.033	0.029
8	0.334	6.892	48.228	8.673	0.020	0.043	0.098
9	0.365	8.052	64.480	10.812	0.043	0.054	0.254
10	0.395	9.273	84.914	13.345	0.080	0.067	0.559
11	0.425	10.545	110.462	16.342	0.130	0.083	1.094
12	0.454	11.800	140.415	19.699	0.192	0.101	1.896
13	0.483	13.091	176.868	23.630	0.267	0.123	3.069
14	0.511	14.426	222.397	28.354	0.357	0.149	4.771
15	0.538	15.763	278.262	33.952	0.461	0.181	7.129
16	0.564	17.070	344.881	40.438	0.575	0.219	10.205
17	0.588	18.350	425.849	48.117	0.703	0.266	14.217
18	0.613	19.629	524.068	57.250	0.843	0.323	19.316
19	0.638	20.878	643.785	68.179	0.996	0.393	25.706
20	0.659	22.020	780.140	80.442	1.152	0.475	33.068
21	0.677	23.119	947.159	95.263	1.321	0.579	42.061
22	0.694	24.196	1158.476	113.809	1.510	0.718	53.162
23	0.712	25.171	1411.559	135.812	1.711	0.897	65.902
24	0.726	26.066	1723.454	162.727	1.913	1.139	80.177
25	0.735	26.765	2060.105	191.601	2.089	1.431	93.810
26	0.745	27.358	2426.795	222.940	2.243	1.794	106.477
27	0.754	27.840	2795.757	254.400	2.360	2.216	116.835
28	0.763	28.325	3260.879	293.976	2.450	2.854	126.123
29	0.773	28.812	3864.764	345.266	2.472	3.929	131.650
30	0.782	29.300	4679.512	414.354	2.330	6.122	126.949
31	0.789	29.569	5362.202	472.152	2.112	9.321	117.396

Table 25 # Computation of waiting times of individual classes when preemptive resume priority is followed (265 Slots)

4.7. Implications and conclusion

Given that software maintenance is one of the most important and costly phases of the software product life cycle, managers need to explore newer ways of improving productivity in maintenance operations. Even though it is commonly believed that debugging is an important and resource intensive activity in software development life cycle, it has been labeled an understudied problem not only in computer science – as cited in Liberman (1997) – but also from a management science perspective. To the best of our knowledge, this is the first work to apply management science principles to manage debugging tasks in a software maintenance environment. From a managerial perspective, the implications of this work are the following:

First, we suggest that managers may cut off working on a bug and still not greatly reduce their chance of successful resolution. Our computational study suggests time frames that one may pick for imposing the cut off threshold. We show that the probability of successfully resolving the tasks reduce with time, and imposition of threshold cut off can result in reducing the overall bug queues and minimize resolution delays without having a significant impact on the overall probability of successfully resolving bugs. From a service provider's perspective, this can not only improve the utilization of current engineering resources, but also improve customer satisfaction by reducing waiting times.

Second, the methodology can provide a good estimate of engineering capacity required in debugging environments. The fact that the estimates provided by the model closely matched the raw data in the example firm, lend credence to the approach adopted.

Third, when individuals are working on bugs that come from multiple populations, we demonstrate that there may be benefits that can be gained if the nature of population is taken into account particularly when the environment is capacity constrained. For example, the section on multiple group analysis investigates capacity constrained situations wherein the rates of successful resolution can be maximized.

Bennett (1996) quotes: “The inability to undertake maintenance quickly, safely and cheaply means that for many organizations, a substantial applications backlog builds up.” (pp. 674) This phenomenon is clearly seen in our analysis. However, our analysis shows that one of the reasons for the phenomenon can be the nature of the process. We show that alternative policies of cutting off work beyond certain time may be a viable alternative to working on tasks that take unreasonably long.

Finally, in this work, our objective is to study the performance of the service system, we abstract away from the question “what introduces heterogeneity in the population of bugs?[31]”However, the abstraction does not obscure the validity of the model (beta-geometric hazard). The model predicts both the combined data and sub classifications of the population very well.

The idea of suggesting a threshold cut-off is not to suggest that bugs should be hidden, but we endeavor to understand the queueing tradeoffs in a debugging environment. Implementing such a policy in practice may be challenging. One of the ways such a policy can be implemented is that bugs could be flagged after *x* time periods for review by senior technical experts on the current situation to take an expedited decision on the current state of the bug.

From the standpoint of generalization there are two aspects that need to be considered. First, can we use this approach to study other bug fixing situations? We believe

[31] Examples of work that investigate sources of heterogeneity are Kitchenham et al. (1999) and Kemerer and Slaughter (1997)

that this is possible given that the process of bug fixing is very much the same in most situations. However, this needs to be verified through further studies. Second, what are the other scenarios where this model may be applicable? We believe that such models can be applied to diverse environments/processes that may have the characteristics of decreasing probability of success with time. For example, product development environments where research has revealed the presence of a split population in the emergence of dominant designs with the passage of time (Srinivasan et al. 2006). Similar models with respect to developing new product or a project could be conceived where there is an uncertainty to completion of such projects and require a resource commitment from a management perspective. In the financial services industry, several researchers have used survival models for investigating the factors that affect the survival time of the loans (Shumway 2001; Roszbach 2004; Carling et al. 2001). Models used in this chapter can be adapted to examine the capacity of backend processes that may be needed in these environments.

CHAPTER 5

5. Conclusion and Future Research

Offshore outsourcing is today common in many industries, and one that is likely to grow over time as the legal, political, and technological barriers to global commerce are further dismantled. Despite this growth, the practitioner industry has cited dissatisfaction with the services that are delivered from such settings. This dissertation sheds light on managing software service operations by considering two different, but intricately linked facets of software operations – internal and external.

The second chapter of the dissertation titled "*Managing Outsourced Offshore projects: Antecedents of Project Performance and Customer Satisfaction*" investigated the external facet of software operations. In this chapter, we adopted a multidisciplinary perspective to investigate how project management, team stability, communication effectiveness and other variables drive project performance and customer satisfaction. Using a structural equations model, first, we show that team stability, project management, communication effectiveness, and project performance impact overall customer satisfaction. Next, we demonstrate that contextual variables such as the nature of work (maintenance & development versus testing projects), the class of software (application software versus system software), and project duration (high age versus low age projects) moderate the overall impact of the antecedents on project performance and customer satisfaction.

In the third and fourth chapters of the dissertation, we investigate the "internal facet" of a software services organization. Focusing on software maintenance projects, we investigate antecedents of engineer productivity and methods to manage the process better with the aim of improving individual productivity. In chapter 3 of this dissertation titled "*Individual Learning and Productivity in a Software Maintenance Environment: An Empirical Analysis*" we investigate the determinants of individual learning in settings where individuals work on a variety of tasks. Our results in this work show that, first, performing a greater variety of tasks improves individual productivity. However, somewhat paradoxically, this increase in productivity involves an initial loss of productivity as the individual invests effort in learning new tasks to develop higher competency. Second, we also find that cumulative variety of tasks improves productivity. However, variety benefits more when the nature of experience is intensive. Third, we find that striving to gain a balance between intensiveness and extensiveness of experience leads to the highest productivity. Fourth, our results show that more experienced individuals can handle new tasks faster as compared individuals with less experience reflecting the transferable nature of experience accumulated. Fifth, individuals that belonged to a larger team were more productive than individuals that belonged to a smaller team reflecting evidence of learning in an environment where the individuals need not be performing the same task but can learn through collocation and informal collaboration and finally, we find empirical evidence of the impact of turnover and present nuances on the moderating effect of team size on the impact of turnover on productivity.

Finally, in chapter 4 of the dissertation, titled "*Resource Allocation in Software Maintenance*," we use the inherent characteristics of the debugging process to show that productivity of software engineers can be improved by cutting off working on bugs after a certain time period. We show using actual data that as software defects (bugs) stay in the system longer, the probability of successfully resolving them significantly reduces. Further, we use this insight to estimate required engineering resources to manage debugging efforts. Finally, we explore the cut off policies based on threshold time without finding a successful resolution. Our analysis in this work shows that such policies minimally impact rates of successful resolution, and reduce waiting times for incoming bugs in the system. Through a computational study, we explore how such policies may improve overall productivity and enable efficient utilization of engineering resources. Together these three essays can contribute to the better management of software service operations and enable managers to improve productivity, and consequently, customer satisfaction.

Extensions to Dissertation

This dissertation can be extended on several fronts to understand the overall issue of offshore outsourcing better.

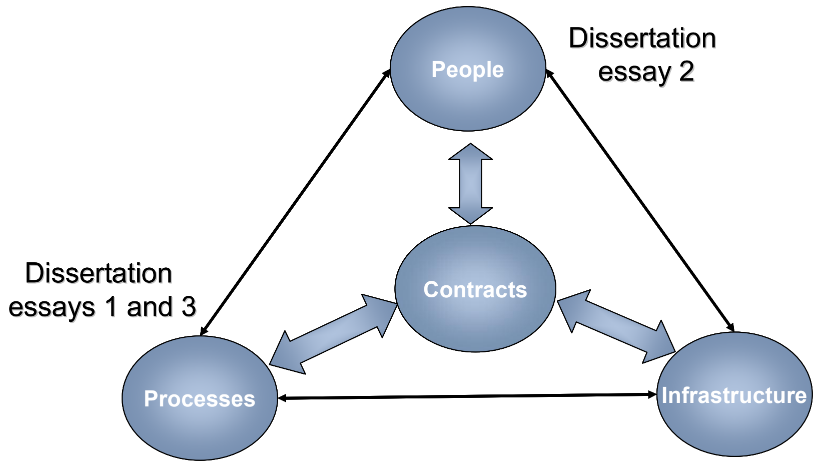

Figure 15 # Framework for future research

Figure 15 gives a framework on which the current dissertation is built and will also provide a framework for future research. The underlying idea of the framework is the firms have to have effective integration of people, process and infrastructure and have this integration in harmony with the contractual agreements to achieve higher performance. The first and the third essays – chapter 2 and chapter 4 – of this dissertation investigate the process level perspectives of managing software operations. For example, in chapter 2 we investigate how processes of project management, communication and team stability need to be managed to enable better project performance and customer satisfaction. Similarly in chapter 4 we exploit the characteristics of the process (the probability of successful resolution reaches zero with increasing time) to investigate opportunities to make improvements in the efficiency of the process. The second essay of the dissertation – chapter 3 – investigates aspects of how people can be trained to increase their performance by designing task allocation mechanisms.

Based on the above framework, several extensions are possible that would be worth exploring. At the process level, the first essay can be extended by looking at the following framework. The key antecedents of customer satisfaction can be divided into (a) Outcome based antecedents, refers to measurable outcomes such as timeliness and quality as considered in the current setting (b) Procedure related antecedents, refers to the means adopted for the delivery of the product (such as project management) and finally (c) Experience based antecedents, refer to issues that can contribute to the experiences of the client in the delivery process.

Considering all the three antecedents can provide a holistic view of offshore project delivery. Table 26 outlines some of the key issues that can be addressed in any one of the three classes of antecedents. For example, in the context of output, one could incorporate and measure the intellectual contributions of the software development teams to the customer's knowledge base. In the context of procedure based antecedents, one could include clients satisfaction with the vendor's staffing, training and resource management, intellectual property protection and quality related processes. These may have a direct impact on the operational efficiency at the vendors end. Similarly, some of the experience related antecedents may include the level of commitment and empathy shown by the vendor's project teams with the client teams, the cultural compatibility between the client and the vendor staff, the level of 'ownership' of the work shown by the offshore team and their involvement in providing proactive suggestions on various aspects of the project. Further, the interplay between project performance and the nature of financial contracts signed between the customer and the service provider can be studied (e.g., Gopal et al. 2003). Finally, issues related to competition and market performance in the context of offshore outsourcing can be investigated in this context.

Outcome Based Antecedents	Procedure based Antecedents	Experience based Antecedents
a) Quality of deliverable b) Timeliness of deliverables c) Adherence to service level agreement parameters d) Process and product innovations	a) Was the project managed well? b) Are the processes appropriate? c) Does the vendor have clear guidelines for handling issues that the engagement may face? d) Does the vendor have adequate mechanisms to fill in vacant slots in projects e) What are the process improvement procedures in place? f) How are procedures on intellectual property protection enforced	a) Do the vendors engineers communicate well? b) Is there culture compatibility between offshore and onsite teams? c) How does the vendor handle crisis situations? d) Do they proactively raise issues and bring problems to light? e) Ownership of products and services by the vendor

Table 26 # Outcome, procedure and experience based antecedents in offshore projects

In the context of learning (people related aspects of the framework in Figure 14), the following issues appear to particularly merit research attention. First, how should the learning pattern for an individual be crafted to maximize the productivity of the team and the firm? This pattern, which would ideally be sensitive to the existing experience of the individual and the knowledge sharing pathways within the team, would lay out how the individual's exposure to task specialization and task variety would vary across time. This approach would also better integrate the training and knowledge sharing literatures. Further, the pattern of learning across projects can influence the adoption of higher-level problem solving strategies. For example, learning by trial and error and selectionism constitute two distinct problem solving strategies in the product development context (Sommer and Loch 2004). These approaches involve different levels of exposure to variety, and hence different learning outcomes.

Second, the roles of exposure to variety and specialization in driving learning can be connected more tightly with the concept of job design. One can conceptualize job design as detailing the choice and sequences of tasks carried out by an employee, the linkages between these tasks and those of co-workers, and the incentive and motivational system that encourages employees to input effort into the tasks. Next, the issue of how the concept of job design can be expanded to accommodate the correct balance between task specialization and exposure to variety towards enhancing employee learning and productivity becomes relevant in this context.

Third, we can investigate the impact of "Recency Effect" – how recent is the exposure – on the overall rates of learning and the role of recent exposure to variety. For example Nembhard and Osothsilp (2001) compare different types of forgetting models in their work. Some of these models can be applicable in the context of knowledge workers. One such application was by Shafer et al. (2001) who investigated forgetting effects on assembly line workers by empirical means. In a similar environment McCreery et al. (2001) find that for complex tasks forgetting effects in assembly line environment may be high. Such effects may be worth investigating in our setting.

Fourth, another potential area of research in the context of variety is to examine degree of interrelatedness of tasks on learning outcomes. Even though the tasks performed by the individuals in our study are related, the degree of interrelatedness and its effect on productivity needs to be better understood. This is because relatedness in task can be a continuum depending on the number of common skills between two different tasks since the time taken to learn a task is proportional to the total number of different skills required to learn the task (Jovanovic and Griliches 1995). Such an understanding may enable us to gain insights on the relationship between task interrelatedness, productivity and task variety. However, given the nature of the current dataset that may not be a feasible option.

Finally, the idea of HHEI can be applied to several other situations to explain the influence of variety on productivity in manufacturing operations literature. For example, Ramdas (2003) pointed out that researchers such as Fisher and Ittner (1999) and Macduffie et al. (1996) arrived at opposite conclusions on the impact of product variety on assembly line productivity. Indices such as HHEI could be tried in such contexts to investigate the overall exposure of relative variety levels in this scenario.

Finally, from a process perspective, the work in chapter 4 can be expanded into other areas. First, the work on chapter 4 can also be extended to consider capacity planning and resource allocation in product development environments. This approach is also likely to be consistent with prior research that has revealed the presence of a split population – i.e. a few designs will never see the light of the day – in the emergence of dominant product designs (Srinivasan et al. 2006). Similar models with respect to developing new product or a project could be conceived where there is an uncertainty to completion of such projects and require a resource commitment from a management perspective.

Second, we are currently considering is the idea of dynamic task allocation. These problems are also called online task allocation problems. In this setup, the manager faces the problem of scheduling tasks 'real time' to project members. The key characteristic of this problem is that tasks have to be assigned to engineers at a given time with uncertainty about future arrivals. This problem differs from standard scheduling problems in the sense that the after each allocation the experience of the engineer is updated and evolves and thus this also impacts the myopic allocation decisions. Understanding the effects of such myopic real time task allocation on overall productivity of the team is an interesting problem that can be considered.

Finally, on the role of contracts, one could investigate the role of contracts and incentives on the overall performance of the projects. For example Gopal et al. (2003) investigate the antecedents of fixed bid and time and material projects in organizations. They point to project characteristics that made organizations to sign fixed bid and time and material contracts in offshore projects. However, the nature of people that staff the projects are also governed by the nature of contracts. In this context, questions such as "Do projects perform better when contracts signed are fixed bid or time and material?," "How do the contracts impact the choice of infrastructure, people and process related practices in an outsourced environment?," still need to be answered. Many of these factors may have a direct impact on the performance of such projects. Further, the framework can also provide some insights and directions on evaluating the bids for projects for both the customer and the vendor by examining the linkages between the contracts and the other infrastructure parameters. These factors may be worth investigating as extensions to the dissertation.

APPENDIX

Derivation of the Beta Geometric Distribution

The generic probability of resolution when p is drawn from a distribution with pdf $f(p)$ is given by equation *1*

$$p_t^{'} = \int_0^1 p^t (1-p)^{t-1} f(p) dp \tag{22}$$

When p is drawn from a beta distribution with parameters α and β:

$$f(p) = \frac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p)^{\beta-1},\ \alpha \geq 0\ ,\ \beta \geq 0 \tag{23}$$

Substituting for *f(p)* in equation *22* we get

$$p_t^{'} = \frac{1}{B(\alpha,\beta)} \int_0^1 p^t (1-p)^{t-1} p^{\alpha-1} (1-p)^{\beta-1} dp \tag{24}$$

$$p_t^{'} = \frac{1}{B(\alpha,\beta)} \int_0^1 p^{t+\alpha-1} (1-p)^{(t+\beta-1)-1} dp \ = \ \frac{B(t+\alpha, t+\beta-1)}{B(\alpha,\beta)} \tag{25}$$

The probability of not resolving until period t-1 is given by

$$P(T > t-1) = 1 - \frac{\Gamma(\alpha+\beta)}{\Gamma\alpha\,\Gamma\beta} \sum_{t=1}^{t-1} \frac{\Gamma(\alpha+1)\Gamma(\beta+t-1)}{\Gamma(\alpha+\beta+t)} \tag{26}$$

Using equations *25* and *26* the conditional probability of resolving at time *t* given that the bug has not been successfully resolved until time *t-1* is given by:

$$p_t = P(T = t \mid T > t-1) = \frac{\dfrac{B(t+\alpha, t+\beta-1)}{B(\alpha,\beta)}}{1 - \dfrac{\Gamma(\alpha+\beta)}{\Gamma\alpha\,\Gamma\beta} \displaystyle\sum_{t=1}^{t-1} \frac{\Gamma(\alpha+1)\Gamma(\beta+t-1)}{\Gamma(\alpha+\beta+t)}} \tag{27}$$

Simplifying equation *27* gives us the expression in equation *2*.

Likelihood Function – Beta-Geometric Hazard

In each period the bugs can be classified as successfully resolved or unsuccessfully resolved. The likelihood function for each period can be written as:

$$P_t(X = m) = p_t^m (1 - p_t)^{n-m} \tag{28}$$

$$\textit{Where } p_t = \frac{\alpha}{\alpha + \beta + t - 1} \tag{29}$$

The likelihood function can be written as:

$$L = \prod_{t=1}^{n} \left(\left(\frac{\alpha}{\alpha + \beta + t - 1} \right)^m \left(\frac{\beta + t - 1}{\alpha + \beta + t - 1} \right)^{n-m} \right) \tag{30}$$

An Alternative Approach to calculating the Beta-Geometric Hazard

An alternative approach to calculating the hazard that we will frequently use for our purposes is attributed to Weinberg and Gladen (1986). Equation *29* can be decomposed into a series of cycle specific hazard rates.

$$p_t^{'} = P(T = t \mid T > t - 1) \prod_{j=1}^{t-1} P(T > j \mid T > j - 1) \tag{31}$$

If we denote p_j as the hazard rate then the equation *31* can be written as

$$p_t^{'} = \Pr(T = t) = p_t \prod_{j=1}^{t-1} (1 - p_j) \quad \textit{and} \tag{32}$$

$$\Pr(T > t) = \prod_{j=1}^{t} (1 - p_j) \tag{33}$$

Trinomial Model

This model draws from the beta-geometric model. In this model bugs can be categorized into three distinct groups. The first group is one that is resolved in each period. The second group is the one that is classified as cannot be resolved. Finally, the third group is the carried forward into the next period on which no decision is made. We assume that in each trial, the probability of resolution is drawn from a beta-geometric distribution. The Likelihood function for this case is

$$L = ((1-\theta)p)^m \theta^s ((1-\theta)(1-p))^{n-m-s} \tag{34}$$

Where θ is the population that cannot be resolved, p is given by equation *2*. The reasoning for the likelihood equation is as follows. In each period a bug is resolved with a probability $(1-\theta)p$ because the bug does not belong to the "irresolvable" category, is declared "irresolvable" with probability θ and finally can be carried forward with a probability $((1-\theta)(1-p))$.

Thus the resulting likelihood function for which parameters need to be estimated will be:

$$L = \left((1-\theta)\left(\frac{\alpha}{\alpha+\beta+t-1} \right) \right)^m \theta^s \left((1-\theta)\left(1-\frac{\alpha}{\alpha+\beta+t-1} \right) \right)^{n-m-s} \tag{35}$$

Nozaki and Ross (1978) approximation

$$W(G) \approx \frac{\lambda^n E[X(a)^2](E[X(a)])^{n-1} p_0(\rho)}{2(n-1)!(n-\lambda E[X(a)])^2} \tag{36}$$

where $\rho = \lambda E[X(a)]/n$ and $p_0(\rho) = \left(\sum_{i=1}^{n-1} \frac{(n\rho)}{i!} + \frac{(n\rho)^n}{n!(1-\rho)} \right)^{-1}$

Boxma et al. (1979) approximation

$$W(G) \approx \left(W(M) + (2W(D) - W(M)) \left(\frac{E[X(a)^2]}{\gamma_1 - n - 1} \right)(m-1) \right)^{-1} \left(1 + \left(\frac{\sigma_{X(a)}}{\mu_{X(a)}} \right)^2 \right) W(M)W(D) \tag{37}$$

The value of W(D) (waiting time for a M/D/n priority queue) can be obtained from the approximations suggested by Cosmetatos (1975) where

$$W(D) \approx \frac{1}{2} W(M)[1 + (16n\rho)^{-1}(1-\rho)(n-1)(\sqrt{4+5n} - 2)] \tag{38}$$

and $$\gamma_1 = \frac{\left(1 - \left(\frac{\sigma_{X(a)}}{\mu_{X(a)}} \right)^2 \right) E[X(a)]}{n+1} + \frac{\left(\frac{\sigma_{X(a)}}{\mu_{X(a)}} \right)^2 E[X(a)]}{n} \tag{39}$$

REFERENCES

Abdel-hamed, T. K. 1989. The Economics of Software Quality Assurance: A Simulation based case study. MIS Quarterly 12(3) 395-411.

Abernathy, W. J., K. Wayne. 1974. Limits of the learning curve. Harvard Business Review 52(5) 109.

Amoribieta, I., K. Bhaumik, K. Kanakamedala, A. D. Parkhe. 2001. Programmers abroad: A primer on offshore software development. Mckinsey Quarterly 2

April, A., A. Abran, R. R. Dumke. 2004. Software Maintenance Productivity measurement: How to assess the readiness of your organization, 14th International workshop on Software Measurement

Apte, U. M., R. O. Mason. 1995. Global disaggregation of information-intensive services. Management Science 41(7) 1250-1263.

Apte, U., M., M. Sobol, G., S. Hanaoka, T. Shimada, T. Saarinen, T. Salmela, A. P. J. Vepsalainen. 1997. IS outsourcing practices in the USA, Japan and Finland: a comparative study. Journal of Information Technology (Routledge, Ltd.) 12(4) 289-304.

Argote, L. 1993. Group and Organizational Learning Curves: Individual, System and Environmental Components. British Journal of Social Psychology 32(1993) 31-51.

Argote, L. 1999. Organizational Learning: Creating, Retaining and Transferring Knowledge. Kluwer Academic Publishers Massachusetts.

Argote, L., S. L. Beckman, D. Epple. 1990. The Persistence and transfer of learning in Industrial Settings. Management Science 36(2) 140-155.

Argote., L., B. McEvily., R. Reagans. 2003. Managing Knowledge in Organizations: An Integrative Framework and Review of Emerging Themes. Management Science 49(4) 571–582.

Arora, A., V. S. Arunachalam, J. Asundi, R. Fernandes. 1999. The Indian Software Industry, Report submitted to the Alfred Sloan Foundation, Carnegie Mellon University from http://www.heinz.cmu.edu/project/india

Asher, H. 1956. Cost-Quantity Relationships in the Airframe Industry. Rand Corp. Santa Monica, CA.

Balasubramanian, S., P. Konana, N. M. Menon. 2003. Customer Satisfaction in Virtual Environments: A Study of Online Investing. Management Science 49(7) 871-889.

Baloff, N. 1971. Extension of the Learning Curve - Some empirical Results. Operational Research Quarterly 22(4) 329-340.

Banker, R. D., S. M. Datar, C. F. Kemerer, D. Zweig. 2002. Software Errors and Software Maintenance Management. Information Technology and Management 3 25-41.

Banker, R. D., S. M. Datar, C. F. Kemerer. 1991. A model to evaluate variables impacting the productivity of software maintenance projects. Management Science 37(1) 1-18.

Banker, R., D. Gordon, S. Slaughter. 1998. Software development practices, software complexity, and software maintenance performance. Management Science 44(4) 433-451.

Banker, R., S. Slaughter. 1997. A field study of scale economies in software maintenance. Management Science 43(12) 1709-1725.

Barki, H., S. Rivard, J. Talbot. 1993. Toward an Assessment of Software Development Risk. *Journal of Management Information Systems* **10**(2) 203-225.

Batt, R., P. Osterman. 1993. A National Policy for Workplace Training: Lessons from State and Local Experiments, Economic Policy Institute, Washington, D.C.

Bendapudi, N., R. P. Leone. 2002. Managing Business-to-Business Customer Relationships Following Key Contact Employee Turnover in a Vendor Firm. Journal of Marketing 66(2) 83-101.

Bennett, K.H. "Software Evolution: past, present and future.," Information and Software Technology (39:11), 1996, pp. 673-680.

Berry, L. L., A. Parasuraman, V. A. Zeithaml. 1988. The service-quality puzzle. Business Horizons 31(5) 35-43.

Berry, L. L., V. A. Zeithaml, A. Parasuraman. 1985. Quality counts in services, too. Business Horizons 28(3) 44-52.

Berry, P. a. Z. 1985. Quality Counts in Services, Too. Business Horizons(May-June) 44-52.

Bishop, J., S. Kang. 1996. Do Some Employers Share the Costs and Benefits of General Training. Center for Advanced Human Resource Studies, Cornell University: Ithaca, NY.

Black, J. S., M. Mendenhall. 1990. Cross-Cultural Training Effectiveness: A Review and a Theoretical Framework for Future Research. Academy of Management Review 15(1) 113-136.

Bland, V. 2006. IT skills shortage still causing pressure, Vol. 2006. The New Zealand Herald

Boehm, B. W. 1981. Software Engineering Economics. Prentice Hall

Boehm, B. W. 1989. Software Risk Management. IEEE Computer society press

Boh, W. F., Slaughter, S. A. and Espinosa, J. A. (2006) Learning from experience in software development: A multi-level analysis. Management Science. Forthcoming

Bolton, R. N. 1998. A Dynamic Model of the Duration of the Customer's Relationship With a Continuous Service Provider: The Role of Satisfaction. Marketing Science 17(1) 45-66.

Bolton, R. N., K. N. Lemon. 1999. A Dynamic Model of Customers' Usage of Services: Usage as an Antecedent and Consequence of Satisfaction. Journal of Marketing Research 36(2) 171-186.

Bondi, A.B., and Buzen, J.P. 1984. "The response times of priority classes under preemptive resume in M/G/m queues," ACM SIGMETRICS Performance Evaluation Review 12(3) 195-201.

Boomsma, A., J. J. I. E. Hoogland, Structural equation models: Present and future. A Festschrift in honor of Karl Jöreskog (pp. 139-168). Chicago: Scientific Software International (Eds.). 2001. The robustness of LISREL modeling revisited. Scientific Software International Chicago.

Boudreau, J., W. Hopp, J. O. McClain, T. L. Joseph. 2003. On the Interface Between Operations and Human Resources Management. Manufacturing & Service Operations Management 5(3) 179–202.

Boxma, O.J., Cohen, J.W., and Huffels, N. "Approximations of the Mean Waiting Time in an M/G/s Queueing System," Operations Research (27:6), 1979, pp. 1115-1128.

Brooks, F. 1995. The Mythical Man-Month: Essays on Software Engineering (second ed.). Addison Wesley

Brown, J. S., P. Duguid. 1991. Organizational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning, and Innovation. Organization Science 2(1) 40-57.

Brown, S. W., T. A. Swartz. 1989. A Gap Analysis of Professional Service Quality. Journal of Marketing 53(2) 92-98.

Bryan, W. L., N. Harter. 1899. Studies on the telegraphic language. Psychological Review 6 345-375.

Burnham, K. P., D. R. Anderson. 2004. Multimodel Inference: Understanding AIC and BIC in Model Selection. Amsterdam Workshop on Model Selection, Amsterdam.

Cappelli, P., N. Rogovsky. 1994. New work systems and skill requirements. International Labour Review 133(2) 205-221.

Carley, K. 1992. Organizational learning and personnel turnover. Organization Science 3(1) 20-46.

Carmel, E., R. Agarwal. 2000. Offshore Sourcing of Information Technology Work by America's Largest Firms, Technical Report, Kogod School. American University: Washington D.C.

Carmel, E., P. Tija. 2005. *Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce*. Cambridge University Press Cambridge.

Chapman, R. J. 1998. The role of system dynamics in understanding the impact of changes to key project personnel on design production within construction projects. *International Journal of project Management* **16**(4) 235-247.

Chase, R. B. 1978. Where Does the Customer Fit in a Service Operation? Harvard Business Review 56(6) 137-142.

Clair, B., G. Linden. 2005. Offshoring in the Semiconductor Industry: A Historical Perspective, from http://web.mit.edu/ipc/sloan05/BrownLindenOffshore.pdf

Clifton, D. 2005. Outsourcing documentation development: Assessing the offshore option,, *WinWriters. Inc from http://www.winwriters.com/articles/outsourcing/index.html*

Cohen, W., M. , D. A. Levinthal. 1990. Absorptive Capacity: A New Perspective on Learning and Innovation. Administrative Science Quarterly 35(1) 128-152.

Cohen, M. D. 1991. Individual Learning and Organizational Routine: Emerging Connections. Organization Science 2(1) 135-139.

Cook, P. D., G. Chon-Huat, H. C. Chen. 1999. Service typologies: A state of the art survey. Production and Operations Management 8(3) 318.

Cosmetatos, G.P. "Approximate explicit formulae for the average queueing time in the process M/D/r and D/M/r," Informatics:328-331, 1975,

Crossan., M. M., H. W. Lane., R. E. White. 1999. An Organizational Learning Framework: From Intuition to Institution. The Academy of Management Review 24(3) 522-537.

Curtis, B., H. Krasner, N. Iscoe. 1988. A field study of the software design process for large systems. Communications of the ACM 31(11) 1268-1287.

Czepiel, J. A. 1990. Service encounters and service relationships: Implications for research. Journal of Business Research 20(1) 13-21.

Danaher, P. J., J. Mattsson. 1994. Customer Satisfaction during the Service Delivery Process. European Journal of Marketing 28(5) 5 - 16.

Deephouse, C., T. Mukhopadhyay, D. R. Goldenson, M. I. Kellner. 1996. Software processes and project performance. Journal of Management Information Systems 12(3) 185-203.

Dibbern, J., T. Goles, R. Hirschheim, B. Jayatilaka. 2004. Information systems outsourcing: a survey and analysis of the literature. ACM SIGMIS Database 35(4) 6-102.

Dutton, J. M., A. Thomas. 1984. Treating Progress Functions as a Managerial Opportunity. The Academy of Management Review 9(2) 235-247.

Earl, M. J. 1996. The Risks of Outsourcing IT. Sloan Management Review 37(3) 26-32.

Edmondson, A. C., R. Bohmer, G. P. Pisano. 2001. Disrupted Routines: Team Learning and New Technology Adaptation. Administrative Science Quarterly 46 685-716.

Ehrlich, W. K., T. J. Emerson. 1987. Modelling software failures and reliability growth during system testing, 9th International Conference on Software Engineering 72–82

Eisenstadt, M. 1997. My hairiest bug war stories. Communications of ACM 40(4) 30-37.

Ellis, H. 1965. The Transfer of Learning. Macmillan Company New York.

Engardio, P. 2006. The Future of Outsourcing, Business Week, Vol. January 30, 2006

Estelami, H., R. F. Hurley. 2003. Does Employee Turnover Predict Customer Satisfaction? Report NO:02-121. Marketing Science Institute

Ethiraj, S. K., K. Prashant, M. S. Krishnan, J. V. Singh. 2005. Where do capabilities come from and how do they matter? A study in the software services industry. Strategic Management Journal 26(1) 25-45.

Fiol, C. M., M. A. Lyles. 1985. Organizational Learning. The Academy of Management Review 10(4) 803-813.

Fisher, M. L., C. D. Ittner. 1999. The Impact of Product Variety on Automobile Assembly Operations: Empirical Evidence and Simulation Analysis. Management Science 45(6) 771-786.

Fjeldstad, R. K., W. T. Hamlen. 1983. Application program maintenance study: Report to our respondents, In Proceedings GUIDE 48: Philadelphia.

Garg, A., P. J. Milliman. 1961. The Aircraft Progress Curve Modified for Design Changes. Industrial Engineering 12(1) 23-27.

Garvin, D. A. 1988. Managing Quality New York.

Goel, A. L., K. Okumoto. 1979. Time-dependent error-detection rate model for software reliability and other performance measures. IEEE Transactions on Reliability 29(3) 206-211.

Gopal, A., S. Konduru, M. S. Krishnan, T. Mukhopadhyay. 2003. Contracts in Offshore Software Development: An Empirical Analysis. Management Science 49(12) 1671-1683.

Gopal, A., T. Mukhopadhyay, K. S. Mayuram. 2002. Virtual extension: The role of software processes and communication in offshore software development. Communications of the ACM 45(4)

Gopal, A., K. Sivaramakrishnan. 2005. On Selecting Appropriate Contract types for Offshore Software Projects: The case of Fixed Price versus Time and Materials Contracts, *Working Paper*

Graydon, J., M. Griffin. 1996. Specificity and variability of practice with young children. Perceptual and Motor Skills 83(83–88.)

Grover, V., M. J. Cheon, J. T. C. Teng. 1996. The effect of service quality and partnership on the outsourcing of information systems functions. Journal of Management Information Systems 12(4) 89-117.

Gwinner, K. P., D. D. Gremler, M. J. Bitner. 2005. Relational Benefits in Services Industries: The Customer's Perspective. Journal of the Academy of Marketing Science 26(2) 101-114.

Hale, D. P., D. A. Haworth. 1991. Toward a model of programmers' cognitive processes in software maintenance: a structural learning theory approach for debugging. Journal of Software Maintenance 3(2) 85–106.

Hale, J. E., S. Sharpe, D. P. Hale. 1999. An Evaluation of the Cognitive Processes of Programmers Engaged in Software Debugging. Journal of Software Maintenance: Research and Practice 11 73–91.

Hamilton, B. H., J. A. Nickerson, H. Owan. 2003. Team Incentives and Worker Heterogeneity: An Empirical Analysis of the Impact of Teams on Productivity and Participation. Journal of Political Economy 111(3) 465-497.

Hatch, N. W., D. C. Mowery. 1998. Process Innovation and Learning by Doing in Semiconductor Manufacturing. Management Science 44(11) 1461-1477.

Hausman, J. A. 1978. Specification tests in econometrics. Econometrica 46(6) 1251-1271.

Heskett, J. L., T. O. Jones, G. W. Loveman, W. E. Sasser Jr., L. A. Schlesinger. 1994. Putting the Service-Profit Chain to Work. Harvard Business Review 72(2) 164-170.

Hoeffler, S., D. Ariely, P. West, R. Duclos. 2006. Preference Exploration and Learning: The Role of Intensiveness and Extensiveness of Experience. Working Paper, Kenan-Flagler Business School, UNC.

Höffler, F., D. Sliwka. 2002. Do new brooms sweep clean? When and why dismissing a manager increases the subordinates' performance. European Economic Review 47(5) 877-890.

Hopp, W. J., E. Tekin, M. P. V. Oyen. 2004. Benefits of Skill Chaining in Serial Production Lines with Cross-Trained Workers. Management Science 50(1) 83 - 98.

Hopp, W., M. Oyen. 2004. Agile workforce evaluation: a framework for cross-training and coordination. IIE Transactions 36(10) 919-940.

Huntley, C. L. 2003. Organizational learning in open-source software projects: An analysis of debugging data. IEEE Transactions on Engineering Management 50(4) 485-493.

Huntley, H. 2005. Five Reasons Why Offshore Deals Fail 1-6. Gartner Report ID G00127840

Johnson, M. K., L. Hasher. 1987. Human Learning and Memory. Annual Review of Psychology 38 631-669.

Jovanovic, B., Y. N. Griliches. 1995. A Bayesian Learning Model Fitted to a Variety of Empirical Learning Curves. Brookings Papers on Economic Activity 247-305.

Kaka, N., J. Sinha. 2005. An upgrade for the Indian IT services industry. Mckinsey Quarterly(2005 Special edition) 85-89.

Kaplan, E. H., A. Hershlag, A. H. Decherney, G. Lavy. 1992. To be or not to be? That is Conception! Managing In Vitro Fertilization Programs. Management Science 38(9) 1217-1229.

Kemerer., C.F., and Slaughter, S.A. 1997 "Determinants of Software Maintenance Profiles: An Empirical Investigation," Software Maintenance: Research & Practice (9) 235-251.

Kitchenham, B. A., G. H. Travassos, A. v. Mayrhauser, F. Niessink, N. F. Schneidewind, J. Singer, S. Takada, R. Vehvilainen, H. Yang. 1999. Towards an Ontology of Software Maintenance. *Journal of Software Maintenance: Research and Practice* **11** 365–389.

Kobayashi-Hillary, M. 2005. A passage to India, ACM Queue, Vol. 3

Konana, P., S. Balasubramanian. 2005. The Social-Economic-Psychological model of technology adoption and usage: an application to online investing. Decision Support Systems 39(3) 505-524.

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

Krishnan, M. S., S. Kekre, T. Mukhopadhyay, K. Srinivasan (Eds.). 1998. Managing Variety in Software Features. Kluwer Academic Publications

Krishnan, M. S., T. Mukhopadhyay, C. H. Kriebel. 2004. A Decision Model for Software Maintenance. Information Systems Research 15(4) 396–412.

Krishnan, M. S., V. Ramaswamy. 1999. Customer Satisfaction for Financial Services: The Role of Products, Services, and Information. Management Science 45(9) 1194-1210.

Kulkarni, V. G., S. Sethi. 2005. Optimal Allocation of Effort to Software Maintenance: A Queuing Theory Approach: Working Paper Series (SOM200554), School of Management, The university of Texas at Dallas.

Lapré, M. A., L. N. Van Wassenhove. 2001. Creating and Transferring Knowledge for Productivity Improvement in Factories. Management Science 47(10) 1311-1325.

Lapré, M. A., N. Tsikriktsis. 2006. Organizational Learning Curves for Customer Dissatisfaction: Heterogeneity Across Airlines. Management Science 52(3) 352-366.

Larson, J. R., C. Christensen. 1993. Groups as problem-solving units: Toward a new meaning of social cognition. British Journal of Social Psychology 32(5-30)

Larsson, R., D. E. Bowen. 1989. Organization and Customer: Managing Design and Coordination of Services. Academy of Management Review 14(2) 213-233.

Lee, J.-N., Y.-G. Kim. 1999. Effect of Partnership Quality on IS Outsourcing Success: Conceptual Framework and Empirical Validation. Journal of Management Information Systems 15(4) 29-61.

Lee, J.-N., S. M. Miranda, Y.-M. Kim. 2004. IT Outsourcing Strategies: Universalistic, Contingency, and Configurational Explanations of Success. Information Systems Research 15(2) 110–131.

Lengnick-Hall, C. A. 1996. Customer Contributions to Quality: A Different View of the Customer-Oriented Firm. Academy of Management Review 21(3) 791-824.

Levin, D. Z. 2000. Organizational Learning and the Transfer of Knowledge: An Investigation of Quality Improvement. Organization Science 11(6) 630-647.

Levinthal, D. A., J. G. March. 1993. The Myopia of Learning. Strategic Management Journal 14(Special Issue) 95-112.

Liberman, H. 1997. The Debugging Scandal and What to do about it. Communications of ACM 40(4) 27-29.

Lientz, B.P., Swanson, E.B., and Tompkins, G.E. 1978 "Characteristics of application software maintenance," Communications of the ACM 21(6) 466-471.

Lievens, A., R. K. Moenaert. 2000. New service teams as information-processing systems: Reducing innovative uncertainty. Journal of Service Research : JSR 3(1) 46.

Littlewood, B. 1980. Theories of software reliability- How good are they and how can they be improved. IEEE transactions on software engineering 6 489-500.

Littman, D. C., J. Pinto, S. Letovsky, E. Soloway. 1987. Mental Models and Software Maintenance. J . Systems and Software 7 341-355.

Loch, C. H., B. A. Huberman, S. T. Stout. 2000. Status Competition and Performance in Work Groups. Journal of Economic Behavior & Organization 43 35 – 55.

Loch, C. H., D. C. Galunic, S. Schneider. 2006. Balancing Cooperation and Competition in Human Groups: The role of Emotional Algorithms and Evolution. Managerial and Decision Economics 27 217-233.

MacCallum, R. C., M. W. Browne, H. M. Suguwara. 1996. Power Analysis and Determination of Sample Size for Covariance Structure Modeling. Psychological Methods 1(2) 130-149.

March, J. G. 1991. Exploration and Exploitation in Organizational Learning. Organization Science 2(1) 71-87.

McCreery, J. K., L. J. Krajewski, G. K. Leong, P. T. Ward. 2004. Performance implications of assembly work teams. Journal of Operations Management **22**(4) 387-412.

McDougall, P. 2005. India, Canada, And China Are Top Outsourcing Destinations, Information Week

McEachern, C. 2005. A Look Inside Offshoring: Customers are less satisfied with offshore service providers, VARBusiness

Mincer, J. 1962. On-the-Job Training: Costs, Returns, and Some Implications. The Journal of Political Economy 70(5) 50-79.

Narayanan, S., S. Balasubramanian, J. M. Swaminathan. 2006. Individual Learning and Productivity in Software Maintenance Environment: An Empirical Analysis, Working Paper: Kenan-Flagler Business School, University of North Carolina at Chapel Hill.

Narayanan, S., S. Balasubramanian, J. M. Swaminathan. 2006. Managing Offshore Software Projects: Antecedents of Project Performance and Customer Satisfaction, Working Paper: Kenan-Flagler Business School, University of North Carolina at Chapel Hill.

NASSCOM. 2005. Strategic review 2005: The IT industry in India. National association for software and service companies: New Delhi, India.

Nembhard, D. A., N. Osothsilp. 2001. An Empirical Comparison of Forgetting Models. IEEE transactions on Engineering Management **48**(3) 283-291.

Newell, A., Simon, H. A. 1972. Human Problem Solving. Englewood Cliffs, NJ: Prentice-Hall

Nidumolu, S. R. 1995. The effect of Coordination and Uncertainty on software project performance: Residual performance risk as an intervening variable. Information Systems Research 6(3) 191-217.

Nidumolu, S. R., M. R. Subramani. 2003. The matrix of control: Combining process and structure approaches to software development. Journal of Management Information Systems 20(3) 159-196.

NIST. 2002. Software Errors Cost U.S. Economy $59.5 Billion Annually, Vol. 2006. National Institute of Standards and Technology

Nitsch, T. R., J. M. Swaminathan. 2006. Managing Product Variety in Automobile Assembly: The Importance of the Sequencing Point. Working Paper

Nosek, J. T. and P. Palvia, "Software Maintenance Management: Changes in the Last Decade," J. Software Maintenance, 2, 3 (1990), 157-174.

Nozaki, S.A., and Ross, S.M. "Approximations in Finite-Capacity Multi-Server Queues with Poisson Arrivals," Journal of Applied Probability (15:4), 1978, pp. 826-834.

Nunnally, J. C. 1978. Psychometric Theory (2nd ed. ed.). McGraw-Hill New York.

Oliva, R., J. D. Sterman. 2001. Cutting Corners and Working Overtime: Quality Erosion in the Service Industry. Management Science 47(7) 894-914.

Olsson, U. H., T. Foss, S. V. Troye, R. D. Howell. 2000. The Performance of ML, GLS, and WLS Estimation in Structural Equation Modeling Under Conditions of Misspecification and Nonnormality. Structural Equation Modeling: A Multidisciplinary Journal 7(4) 557-595.

Overby, S. 2003. The Hidden Costs of Offshore Outsourcing, CIO

Paas, F., J. Van Merrenboer. 1994. Variability of worked examples and transfer of geometrical problem solving skills: A cognitiveload approach. J. Ed. Psych. 86 122–133.

Parker, S. K., R. M. Skitmore. 2005. Project management turnover: causes and effects on project performance. Project management turnover: causes and effects on project performance 23(3) 205-214.

Petersen, M. A. 2005. Estimating Standard Errors in Finance Panel Data Sets: Comparing Approaches. NBER Working Paper No. 11280
Powell, W. W., K. Snellman. 2004. The Knowledge Economy. Annu. Rev. Sociol. 30 199-220.

Pressman, R. S. 2005. Software Engineering: A Practitioner's Approach. McGraw Hill

Quinn, J. B. 1992. *Intelligent enterprise : a knowledge and service based paradigm for industry*. Free Press New York.

Rajlich, V. 1999. Software Change and Evolution. Published in Lecture Notes in Computer Science LNCS 1725, Springer Verlag 186 - 199.

Ramdas, K. 2003. Managing Product Variety: An Integrative Review and Research Directions. Production and Operations Management 12(1) 79-101.

Reagans, R., L. Argote, D. Brooks. 2005. Individual Experience and Experience Working Together: Predicting Learning Rates from Knowing Who Knows What and Knowing How to Work Together. Management Science 51(6) 869-881.

Reber, A. 1989. Implicit learning and tacit knowledge. J. Experiment.Psych. 118 219–235.

Reed, A. 2005. Software Team Turnover: Why Developers Leave (And What You Can Do About it), http://www.developerdotstar.com/mag/articles/PDF/DevDotStar_Reed_DeveloperTurnover.pdf ed., Vol. 2006. Developer.

Rogers, W. H. 1993. Regression standard errors in clustered samples. Stata Technical Bulletin 13: 19–23. Reprinted in Stata Technical Bulletin Reprints, vol. 3, 88–94.

Rus, I., M. Lindvall. 2002. Knowledge management in software engineering. IEEE Software 19(3) 26-38.

Rust, R. T., J. J. Inman, J. Jia, A. Zahorik. 1999. What You Don't Know About Customer-Perceived Quality: The Role of Customer Expectation Distribution. Marketing Science 18(1) 77-93.

Sabherwal, R. 2003. The evolution of coordination in outsourced software development projects: a comparison of client and vendor perspectives. Information and Organization 13(3) 153-202.

Scacchi, W. 1994. "Understanding Software Productivity." Advances in Software Engineering and Knowledge Engineering, 4: 37-70.

Schilling, M. A., P. Vidal, R. E. Ployhart, A. Marangoni. 2003. Learning by Doing Something Else: Variation, Relatedness, and the Learning Curve. Management Science 49(1) 39-56.

Schmenner, R. W. 1986. How can service businesses survive and prosper. Sloan Management Review 3 21-32.

Segars, A. H. and V. Grover (1993). "Re-Examining Perceived Ease of Use and Usefulness: A Confirmatory Factor Analysis." MIS Quarterly **17**(4): 517-525.

Sethi, V., W. R. King. 1994. Development of Measures to Assess the Extent to Which an Information Technology Application Provides Competitive Advantage. Management Science 40(12) 1601-1627.

Shafer, S. M., D. A. Nembhard, M.V. Uzumeri. 2001. The Effects of Worker Learning, Forgetting, and Heterogeneity on Assembly Line Productivity. Management Science **47**(12) 1639 -1653.

Siemsen, E., S. Balasubramanian, A. V. Roth. 2006. Incentives that Induce Effort, Helping and Knowledge Sharing in Workgroups. Working Paper

Simon, H. A. 1979. The Sciences of the Artificial. Cambridge, MA: MIT Press. 2nd ed.

Simon, H. A. 1990. Invariants of Human Behavior. Annual Review of Psychology 41 1-20.

Simon, H. A. 1991. Bounded Rationality and Organizational Learning. Organization Science 2(1) 125-134.

Sitkin, S. B., K. M. Sutcliffe, R. G. Schroeder. 1994. Distinguishing Control from Learning in Total Quality Management: A Contingency Perspective. Academy of Management Review 19(3, Special Issue: "Total Quality") 537-564.

Slocum, J. W., H. Sims. 1980. Typology for integrating technology organization and Job design. Human relations 33(193-212)

Smith, A. 1776. The Wealth of Nations. University of Chicago Press, Chicago.

Sommer, S. C., C. H. Loch. 2004. Selectionism and Learning in Projects with Complexity and Unforeseeable Uncertainty. Management science 50(10) 1334-1348.

Srinivasan, Raji, Gary L. Lilien and Arvind Rangaswamy (2006), "The Emergence of Dominant Designs" Journal of Marketing, 70(April), 1-17.

Stewart, D. M. 2003. Piecing together Service Quality: A framework for Robust Service. Production and Operations Management 12(2) 246-266.

Suchindran, C. M., P. A. Lachenbruch. 1975. Estimates of Fecundability from a Truncated Distribution of Conception Times. DemoFigurey 12(2) 291-301.

Sudhakar, G. P. 2002. Why do young engineers hop companies?, December 17 ed., Vol. 2006. The Hindu

Swaminathan, J. M., H. L. Lee. 2003. Design for Postponement. OR/MS Handbook on Supply Chain Management: Design, Coordination and Operations, edited by Steve Graves and Ton de Kok, Elvesier Publishers.

Swaminathan, J. M., S. R. Tayur. 2003. Models for Supply chains in e-business. Management Science 49(10) 1387-1406.

Teasley, S. D., L. A. Covi, M. S. Krishnan, J. S. Olson. 2002. Rapid software development through team collocation. IEEE Transactions on Software Engineering 28(7) 671 - 683.

Torbiron, L. 1982. Living Abroad. Wiley New York.

Tyre, M., E. von Hippel. 1997. The Situated Nature of Adaptive Learning in Organizations. Organization Science 8(1) 71-83.

van Pul, M. C. 1994. A general introduction to software reliability. CWI Quarterly 7(3) 203-244

Venkatraman, N. 1989. Strategic Orientation of Business Enterprises: The Construct, Dimensionality, and Measurement. Management Science 35(8) 942-962.

Wallace, L., M. Keil, A. Rai. 2004. Understanding Software Project Risk: A Cluster Analysis. Information and Management 42(1) 115-125.

Weinberg, C. R., B. C. Gladen. 1986. The Beta-Geometric Distribution Applied to Comparative Fecundability Studies. Biometrics 42(3) 547-560.

West, S. G., J. F. Finch, P. J. Curran (Eds.). 1995. Structural Equation Models with Non-Normal Variables: Problems and Remedies. Sage Newbury Park, CA.
Whang, S. 1992. Contracting for Software Development. Management Science 38(3) 307-324.

Wirtz, J., J. E. G. Bateson. 1999. Introducing uncertain performance expectations in satisfaction models for services. International Journal of Service Industry Management 10(1) 82-99.

Wooldridge, J. M. 2002. Econometric Analysis of Cross Section and Panel Data. MIT Press Cambridge MA.

Wright, R. L. 2005. Successful IT Outsourcing:Dividing Labor for Success, www.Softwaremag.com

Wulf, G., R. A. Schmidt. 1997. Variability of practice and implicit motor learning. J. Experiment. Psych.: Learn. Memory, and Cognition 23 987–1006.

Youngdahl, W. E., D. L. Kellogg. 1997. The relationship between service customers' quality assurance behaviors, satisfaction, and effort: A cost of quality perspective. Journal of operations management 15(1) 19-32.