2017-01-01

Essays On Sofwtare Development Projects: Impact Of Social And Technological Factors On Project Performance And Co-Diffusion Of Software Sourcing Arrangements

Niharika Dayyala
University of Texas at El Paso, niharika.dayyala@gmail.com

ESSAYS ON SOFWTARE DEVELOPMENT PROJECTS: IMPACT OF SOCIAL AND TECHNOLOGICAL FACTORS ON PROJECT PERFORMANCE AND CO-DIFFUSION OF SOFTWARE SOURCING ARRANGEMENTS

NIHARIKA DAYYALA

Doctoral Program in Business Administration

APPROVED:

Kallol Bagchi, Ph.D., Chair

Peeter Kirs, Ph.D. Co-Chair

Godwin Udo, Ph.D.

Somnath Mukhopadhyay, Ph.D.

Charles Ambler, Ph.D.
Dean of the Graduate School

Dedication

I would like to dedicate my dissertation to:

My mother Vijaya Lakshmi, for her unconditional love.

My brother Siddhartha, for his never-ending support.

To my husband, Sundeep Inti, for always believing in my strengths and for encouraging me throughout this journey.

Finally, to my lovely daughter, Aarna Gracy Inti for teaching me patience and courage. Her smile brings abundant joy to my life.

ESSAYS ON SOFTWARE DEVELOPMENT PROJECTS: IMPACT OF SOCIAL AND TECHNICAL FACTORS ON PROJECT PERFORMANCE AND CO-DIFFUSION OF SOFWTARE SOURCING ARRANGEMENTS

by

NIHARIKA DAYYALA, B.E, M.S

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Accounting and Information Systems

THE UNIVERSITY OF TEXAS AT EL PASO

May 2017

Acknowledgements

I would like to express my sincere gratitude to Dr. Kallol Bagchi, professor, mentor, and primary chair for my dissertation for giving me the freedom to experiment and learn new things. Furthermore, I am grateful to my co-chair, Dr. Peeter Kirs, for being supportive and keeping his door always open for me. This research and dissertation would not have happened without their advice, guidance, knowledge, encouragement, and patience. I would also like to thank Dr. Godwin Udo, and Dr. Somnath Mukhopadhyay for their valuable guidance for this dissertation.

Abstract

Software development is a complicated process which is accomplished by the combined effort of the key elements such as people, processes, and technology. Software development firms are on a constant quest to identify the best practices for managing the key elements to improve their software development project outcomes and to deliver successful software projects. This dissertation aims to study key elements that can influence the software development project performance through three distinctive studies. The first essay of this dissertation investigates the impact of people factors on software project outcomes. Specifically, it examines the impact of team characteristics on the software project outcomes of Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI) level software development projects. Teams and teamwork is a critical component of software development. Software organizations have long since identified the importance of teams and team building in software development. Nevertheless, there is a scarcity of studies on software development team management. Moreover, the role of team characteristics in the software projects that follow the CMM/CMMI practices has been underexplored. The first essay fills the gap by investigating the moderation effects of team characteristics such as team work experience, team turnover (attrition) and team functional heterogeneity on the software development project performance specific to projects that follow the capability maturity process models. The results of the study provide valuable insights into managing the employees such that the software project results in positive outcomes.

The second essay of this dissertation investigates the impact of technological factors on effort and the quality of the software product. Technological factors used in the development process can influence effort estimation process and the quality of the end product. Moreover, the rapid technological advancements in the technologies used, can bring a huge difference in the quality, and the effort estimation of the software product delivered. Through this study, the influence of technical factors on project outcomes is investigated and also the impact of

technological advancement in the technologies is analyzed. This study provides insights about the technologies that can improve the project quality and reduce the development effort.

The third essay considers software sourcing arrangements such as offshoring, on-shoring, in-housing and outsourcing as strategic organizational innovations and analyzes the co-diffusion effects of software sourcing arrangements between 1) on-shoring and offshoring and 2) in-housing and outsourcing. A review of the existing diffusion, co-diffusion, and software sourcing literature revealed that the co-diffusion studies are lacking to understand, how the adoption of one type of software sourcing adoption can influence the adoption of the other. Through the third essay, this gap is filled by performing co-diffusion analysis to provide insights about the current and future trends in the diffusion process of the software sourcing phenomena.

Table of Contents

List of Tables

List of Figures

Chapter 1: Moderation Effects of Team Characteristics in CMM/CMMI Level Software Development Projects

1.1 Introduction

The global business environment has incurred a rapid change due to the emergence of technologies, globalization, and increased dependence on software. This rapid change increased the importance of software development across the world. Software development is a complicated process which is a result of combined effort by key elements such as people, processes, and technology. The three key elements interact with each other to build software systems that are used by various organizations and individuals for running their operations. Software development firms are in constant quest for improving their software development practices to deliver successful software projects. One of such important milestones in improving the success rate of software development is the introduction of the software process improvement (SPI) techniques. Feldt et al. (2012) define SPI is a "systematic approach to enhance the software products and to increase the efficiency and effectiveness of the software process followed by the software development organizations."

SPI research is motivated by the assumption that software development process quality has a direct effect on the software product quality (Cugola & Ghezzi, 1998; Kitchenham & Pfleeger, 1996). Some of the prominent SPI models introduced in the past are Capability Maturity Model (CMM) by the Software Engineering Institute (SEI) in 1995; Capability Maturity Model Integration (CMMI) in 2003; quality management frameworks such as ISO 9001, 2000; performance excellence strategies such as six sigma, etc. These process improvement models aim at ensuring the delivery of software projects with utmost quality, commitment, and efficiency, by making the development process less chaotic, more predictable and manageable in the software development organizations (Magdaleno, Werner, & De Araujo, 2012). They help resolve problems with scheduling and budgeting by keeping the projects on track. Among the many process improvement models, CMM based SPI approaches such as CMM and CMMI have gained much

popularity in software organizations as they provide a way to assess their processes and the overall maturity levels of the organization. They are long-standing, most reported and researched SPI processes in the software field given their introduction approximately 20 years ago (Feldt et al., 2012; Staples & Niazi, 2008). Both the CMM and CMMI aim at software process appraisal and improvement. They are implemented in organizations to help improve project performance and product quality. These methods help an organization's workforce to meet business objectives by assisting them to be efficient and consistent (Lee, DeLone, & Espinosa, 2006). Galin & Avrahami (2006) argued that CMM programs that implement software process improvements can provide the benefits of reaching project milestones on time, reduced development costs, increased productivity, and shorter project duration.

Despite the existence of various quality centric methods and processes improvement techniques to improve the software development process, there are frequent complaints about the high rate of software project failures. Many software projects result in systems that do not function as intended, are not used or not delivered on time (Gaudin, 2003; Gordon, 1999; Y. Lu, Xiang, Wang, & Wang, 2011; Wallace, Keil, & Rai, 2004). Information systems (IS) researchers have reported the software development failure rate on a regular basis. Bloch, Blumberg, & Laartz (2012) stated that software projects run the highest risk of cost and schedule overruns. The Standish group report says that among the 50000 projects developed in 2015: 29 % projects are satisfied, 52 percent are challenged, and 19% are failed. Overall it can be observed that over 71 % of software projects failed due to various reasons. The failed projects are those that exceeded the budget and time offering few features and functions than earlier specified and projects that got cancelled at some point in time in 2015.

Although the reason for the implementation of CMM and CMMI is to reduce the failure rate of software projects; organizations were not successful in eliminating the failures in software development. Cuevas, Calvo-Manzano, & García (2014) state that the equations have changed and rather than the technology and process, the capacity of the software organizations to compete is related directly to the ability to attract, develop, motivate, organize and hold on to people with the

talent required to achieve strategic business targets. Ngwenyama & Nielsen, (2003) state that despite the substantial investments of resources on CMM related SPI initiatives, the failure rate of SPI programs is very high. A considerable number of studies have reported about the failures of organizations that implement CMM and CMMI (Staples & Niazi, 2008). Fuggetta & Di Nitto (2014) note that SPI implementation eliminated many problems, but still some aspects are ill-framed and overlooked. One of the overlooked important aspects of software development is the social aspects of software development. The authors state that improvement in software process is contingent with the proper interplay between human-centered- elements and technological factors and studying them in isolation does not give good results. According to Ngwenyama & Nielsen (2003) the possible explanation for the high rate of failure, in spite of the implementation of CMM and CMMI is the inefficiency to deal with the social aspects of the organization efficiently. In contradiction to the thought of people forty years ago that technology would reduce the necessity of individuals, demand for highly qualified workers has exceeded over the years (Cuevas et al., 2014).

Bloch et al. (2012) identify five main reasons for software project failures. 1) Paying little attention to strategies and stakeholders 2) improper management of projects according to budget 3) inadequate management of technology by critical internal and external talent 4) building weak teams and 5) not focusing on effective core project management practices. Among these, the authors found that failures resulting from improper management of project teams can be detrimental and can lead to more than 50 percent of cost overruns of projects. Thamhain (2004) suggested that among the many factors that drive project performance, human factors and effective team management are critical for project success. Mullaly (2006) identified that the drivers of project management failure are failures in defining processes and roles. As organizations are investing time and resources in critical software projects, investigating and managing the reasons for the failures of software development is paramount (Wallace, Keil, & Rai, 2004). Given the views from previous researchers in software development, it is identified that team related factors are crucial drivers of software project performance and focusing solely on the software process

improvement techniques can result in an imbalance of the software development organizations leading to failed software projects. Given the importance of the social aspects of software development the study focusses on investigating the team characteristics and their impacts on software development project performance.

Software development typically occurs with teams working in an organization. Therefore, the team dynamics can significantly affect the project productivity (Rosson, 1996). According to Lu et al. (2011), the dynamics and interaction of the software development team members are the most important aspect that can influence the IS development project performance. The human factors can directly affect the success and failure of software projects (Guinan, Cooprider, & Faraj, 1998). A substantial percentage of the IS failures are due to social and organizational factors, not just technical factors (Luna-Reyes, Zhang, Gil-García, & Cresswell, 2005). The software organizations quickly understood the rationale that improvements in CMM for the management of development practices cannot come along without significant changes in managing people about the impacts of the social factors. Staples & Niazi (2008) state that human aspects in software development play a paramount role in the success of software development organizations, however, they can be useful only if used efficiently. Some of the team characteristics that can form competent and capable teams are work experience of team members, qualification of team members, and sustainable mix of internal and external resources (Bloch et al., 2012).

Over the past decade, software process research has mainly focused on understanding, evaluating, automating and improving the procedures and policies required to master the complex software development process. Empirical investigations about the role of team characteristics in software development are also very few (Hoegl & Parboteeah, 2007). In the recent times, organizations have identified the importance of workforce, and they are always looking for ways to strengthen and manage their workforce regarding acquiring talent, employee satisfaction, planning ahead for critical positions so that they can fully utilize the capabilities of the workforce (Cappelli, 2008; Yoon & Lim, 2010). However, the role of teams in CMM/CMMI projects has been under-explored. In particular, the moderation effects have not been studied by any other

study. Through this study, software process research takes a step forward by investigating about how team characteristics can impact and moderate the software project outcomes of CMM and CMMI level projects. The paper uses a quantitative approach based on socio-technical theory. Structural equation modeling is adopted to analyze the moderation effects of team characteristics such as work experience of team members, the existence of functional heterogeneity in a team and the turnover in a team on the software project outcomes such as productivity and project elapsed time. The results from the study can provide valuable insights about managing the employees such that the software project may lead to positive outcomes.

1.2 Theoretical Background

The study leverages on the Socio-Technical theory which has been embraced and applied to the design and implementation of the information systems development by several IS researchers till date.

1.2.1 Socio-Technical Theory

The Socio-Technical Theory (STS) by Trist & Bamforth (1951) views an organization as a working system comprising of two interrelated subsystems: ’’ the technical system and the social system. In the STS theory, the “technical subsystem” deals with the processes, techniques, tools, technologies and tasks that are needed to transform the inputs such as raw materials into outputs such as products. The “social subsystem” is concerned with the relationships among people and the attributes of these people such as skills, attitudes, relations, and values and how they work as a whole. The output of a working system is the result of the interaction of the two subsystems (Bostrom & Heinen, 1977). Hanssen (2012) notes that firms can improve their performance regarding productivity and quality of work by considering both the subsystems at the same time. The development processes should be designed such that there is adherence to the technical systems and the social system to bring maximum output (Ravichandran & Rai, 2000). Failure to do so can increase the risks and the systems will not make their expected contribution to the goals

of the organization (Baxter & Sommerville, 2011). The source of the problem for technocentric approaches such as software systems design and development are that they do not adequately consider the complex relations between an organization, people and the systems that support these processes (Baxter & Sommerville, 2011; Goguen, 1999; Norman, 1993).

According to Alter (2013), "a working system is a system in which human participants and machines perform work (processes and activities) using information technology and other resources to produce product/services for internal and external customers." The author also identifies information systems, projects, and supply chains, self-service systems, etc. as work systems that are socio-technical by default. The author also notes that processes are introduced in a working system to produce products in a defined way. Many IS researchers identified software development as a socio-technical phenomenon that attempts to create a new technology product through social and technical interactions by various other researchers. Luna-Reyes et al. (2005) view information systems development as a complex phenomenon embedded in the emergent process of a socio-technical ensemble. Through their research, the authors proposed that IS change is a result of a subtle interplay between technologies, actors, organizational relationships and tasks at multiple levels. Lu et al. (2011) stated that socio-technical theory is concerned about the social and technological factors and that the software development is more of a social event than a technical case. They applied the socio-technical approach to Chinese software development firms and found that the failure rate of IS development projects is very high, and the reasons for the failure is low efficiency and quality of ISD teams.

Fuggetta & Di Nitto (2014) considered software development as a socio-technical system and noted that organizational and human aspects supported by the technology, drive the subsystems. The authors also state that performance improvement in software development does not occur by any of these in isolation. Hu, Zhang, Ngai, Cai, & Liu (2013) emphasized on the STS theory and performed a causality analysis between process performance, product performance, social subsystem risks, and technical subsystem and project management risks. They found that process performance impacts product performance and that all the three critical risk factor can

directly impact the process and the product performances. Wallace et al. (2004) found that the social subsystem influences the technical subsystem, then the project management risk, ultimately impacting the project performance.

The previous research that is driven by socio-technical theory highlights the importance of human factors in software development. However, the software process improvements initiatives such as CMM and CMMI do not address issues related to human resources, finances, marketing, etc. (Clarke & O'Connor, 2012). Given this, it is important to understand the people-relevant factors concerning the software process maturity to improve the software project performance. Therefore, in this study, it is identified that team factors are important, and the team factors concerning the projects developed under the CMM process initiatives are analyzed.

1.2 Literature Review

Software development is a complicated process which requires the process, people, and technology to work together to build successful software projects. Extant literature on software process maturity research is reviewed to find the gaps on the social factors in CMM. IS development research is also examined to identify the team characteristics that can impact the project performance in software development.

1.3.1 CMM, CMMI and the Role of Workforce in Software Development:

The Capability Maturity Model for Software (CMM or SW - CMM) is a framework developed by the SEI in 1995 for assessing the maturity of the software development process of organizations. This framework describes the key elements of an effective software process. The CMM facilitates improvements and optimization in the information systems development process in an organization with a staged five level model built on the proposition that the quality of the software product depends on the quality of the software development and maintenance processes (Paulk, 2009). The five staged model presents recommended practices in the process areas to enhance software development and maintenance capability (Cuenca, Boza, Alemany, &

Trienekens, 2013). An organization with the highest levels of maturity rating excels in attaining more software projects (Silva et al., 2015).

The CMMI was developed in 2000 by the SEI integrating different CMMs (software and product development and management) to provide the organizations with a framework to address issues related to enterprise wide process improvement (Team, 2002). The CMMI shares some similarities in the structure and content with CMM and ISO 9000. It was developed to eliminate the problem of using various CMM's in various disciplines. The CMMI was identified in recent times as resources that are necessary by organizations to win and retain more customers, hence being used by many organizations (Silva et al., 2015).

Software Engineering Institute (http://www.sei.cmu.edu/library/assets/cmmi-overview071.pdf) considers the process, people, and technology as major determinants of a project cost, schedule and quality as shown in Figure 1.1. Among the three determinants, the SEI identifies the process as the glue that ties the people, process, and technology together. Kulpa (2007) states that, while the CMM and the CMMI address the process issues, software methodologies address the technological issues but the people related matters are left unaddressed. The author also states that participating in process improvement activities can be of extra work for employees. Therefore, if the organizational goals do not align with the personal goals, the employees can feel abused and unappreciated resulting in an inadequate workforce performance affecting the output causing the process improvement efforts in organizations to stall or fail. Hence, the author suggests using implementing People-CMM along with CMM as it can help attract, retain and develop individuals and assist them in dealing with the continual changes. The People-CMM employs the process maturity framework of the Capability Maturity Model for Software (more recently called CMMI-DEV) as a foundation for a model of best practices for human resource management (Colomo-Palacios, Casado-Lumbreras, Soto-Acosta, Misra, & García-Peñalvo, 2012). It provides stages through which an organization's workforce and processes evolve. Although people CMM is introduced to assist firms in managing the human aspects, Ngwenyama & Nielsen (2003) note that SW-CMM and P-CMM are contradictory and mutually inconsistent in nature. Although the P-

CMM prescribes team building and participatory culture, the hierarchical structures of CMM work processes with their explicitly defined role responsibilities and strict management control are contradictory to building trust upon which a developmental culture thrives. Thus, implementing only the P-CMM does not eliminate the failures of software organizations.

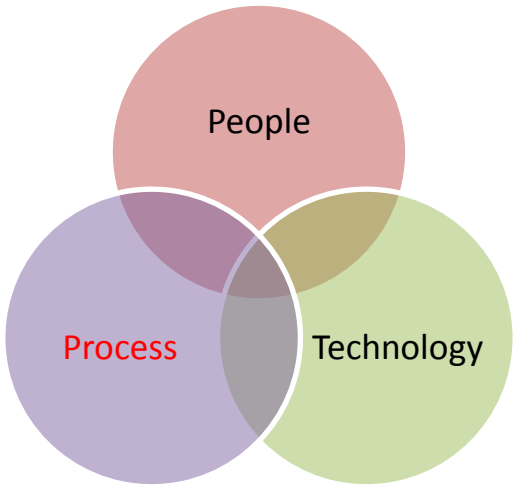


Figure 1.1: People, Process, Technology in Software Development

Another drawback of applying P-CMM is that organizations hire people only when the personnel is required or when the contract is signed. Hence, formal planning measures such as P-CMM for people management is sometimes nonexistent, and the people management responsibility belongs to other organizations (Colomo-Palacios et al., 2012). In many other cases, offshore partners provide the necessary personnel for software development. Therefore it's hard to spread the use of P-CMM to a large extent. Moreover, due to various other reasons, the people factors are given less importance compared to the process factors (Adolph, Kruchten, & Hall, 2012). André, Baldoquín, & Acuña (2011) stated that human resources play a critical role in software project success, but they are the least formalized factor in today's process models.

Cuevas et al. (2014) state that a significant amount of churn occurs in organizations when they grow rapidly and a good process works well only with competent and good people. Due to this reason, few companies have implemented the P-CMM along with the CMMI to improve their competence management along with the process, but they still need additional guidance on this.

CMM research till date focusses mainly on the benefits of implementing Maturity levels in software development process (M. Agrawal & Chari, 2007; Galin & Avrahami, 2006; Harter, Krishnan, & Slaughter, 2000). Adler et al. (2005) interview developers and managers and identified the factors for successful implementation of CMM level projects: encouraging the workforce to be CMM certified, investment of time and resources by the management, involving the staff in defining and refining processes, and facilitating and encouraging the software developers to implement CMM style discipline. El Emam and Koru (2008) found that organizational maturity, methodology, and project management experience would affect project success. Many case studies and research reports have been published on CMM's impact on productivity, cycle time, and quality (Diaz & Sligo, 1997; Haley, 1996; Paulk, 2009). Galin & Avrahami (2006) analyzed 19 papers that were published about CMM and concluded that project performance improved consistently over seven software development performance metrics upon CMM program investments.

1.3.2 Human and Team Factors in Information Systems Development:

In today's business scenario, organizations are competing on two important goals; the first one is to expand their business for products and services and the second one is to acquire talented workforce to help them develop products and services (Cuevas et al., 2014). The success of the business is determined by the skilled workforce that they attain; this effect is very high in case of software development which heavily depends on human capital and their team based activities for the software (Curtis, Hefley, & Miller, 2009). Capretz & Ahmed (2010) state that software is a result of people related activities such as problem-solving, cognitive information processing and social interaction but people are most complicated and non-predictable than computers. Key issues relevant to software engineering boil down to the individuals involved in the software production and their personality traits. Hence, it is important for the organizations to focus more on the human capital management.

Many studies from prior research confirm that IT workforce issues are of key concern to practitioners (Goles, Hawk, & Kaiser, 2008). Goles et al. (2008) note that, to be successful; the IT firms must be able to generate economies of scale by leveraging on the employee skills to satisfy the customers. They also note that the acquiring skilled professionals are an intriguing challenge, and this challenge has compounded because of the continuing growth of outsourcing and the increasing demand for technology solutions. The authors performed an exploratory study and identified the essential skills required by the IT service providers. The authors determined that business domain and project management skills are necessary for the software professionals rather than the technical skills.

Nasir & Sahibuddin (2011) performed a study to identify the critical success factors for software projects. In their study, they identified 26 key success factors for software development and grouped them into three namely, people factors, technology factors and process factors. Based on their examination, from the 26 critical project success factors, they found that seven key success factors (27%) belong to the people factor category, 16 success factors (62%) belong to the process and the remaining three critical factors (12%) belong to the technical factor group.

Capretz & Ahmed (2010) performed a study at the individual level and investigated the relation between personality traits of people and software development using the Myers-Briggs instrument that explains four different personality traits of people in the workplace: extroversion and .introversion, sensing and intuition, thinking and feeling, judging and perceiving. They mapped the software development job requirements such as analysis, design, programming, testing and maintenance and soft skills to the personality traits. The authors found that assigning people with personality types suited for a particular stage of development increase the chances of projects successful outcome. Siau, Tan, & Sheng (2010) performed a qualitative analysis to study the characteristics of good IS development team members and found that cognitive ability, attitude/motivation, and interpersonal communication are essential characteristics of good IS teams.

Kang, Yang, & Rowley, (2006) proposed that software project team success is contingent upon many factors such as team member's coordination, teamwork and shared knowledge. The team success can be moderated by inter-team dynamics, cognitive abilities, cultural contexts, etc. The authors noted that, in software development, team members are responsible for the outcome of projects. The authors performed an empirical study on software development teams and investigated the influences of some member traits on team effectiveness and tested which type of member similarities (cognitive versus demographic) mattered more. The results from their study proved that member cognitive similarities are more important than team member demographic similarities (age, tenure, and gender) for software development team effectiveness.

According to André, Baldoquín, & Acuña (2011) although human factors play a critical role in software project success compared to the process models, people are still the least formalized factor. Project leaders create the teams and assign tasks based on the project managers' experience, constraints (e.g. availability) and skill requirements. This process has to take multiple factors into account. However, very few studies model this process. Most of these are informal proposals focusing on the individual assignment of people to project tasks and do not consider other aspects like team formation as a whole.

Wi, Oh, Mun, & Jung (2009) note that the key to a successful enterprising organization is to collaborate a good leader and competent workers as team members. Thamhain (2004) state that effective teamwork is a critical success factor that provides the competitive advantage to organizations that are usually under pressure to do things faster cheaper and better.

Previous research shows that teams staffed with right people are more likely to be efficient and effective. Nevertheless, there is a paucity of study that examines the important traits of IS professionals in team contexts (Siau et al., 2010). Therefore, it can be inferred that, despite the importance of human resource management in IS, there is a scarcity of studies on software development team management (Kang et al., 2006). Hence, there is a need to explore the effect of team characteristics on software project outcomes (Colomo-Palacios et al., 2012).

1.3.3 Team Characteristics in IS Development:

Project outcomes can be influenced by various team related factors, such as team membership, composition, structure, processes, psychology, tasks, task design, as well as organizational context, resources, structure and environment (Cohen & Bailey, 1997; Kang et al., 2006). The study specifically focusses on the team characteristics such as teamwork experience, team functional heterogeneity, and team turnover. These three team characteristics are identified as important moderators that can impact the project outcomes such as elapsed time and productivity of CMM level software projects.

Team Turnover

Software organizations continually seek out for the high skilled software development professionals due to the rapid growth markets and regions as a result of global software development (Conchúir, Ågerfalk, Olsson, & Fitzgerald, 2009). Due to this, attrition (termed as team turnover in this study) in teams become a unique problem in software industry due to the attractive prospects for trained and experienced software engineers (Gopal, Sivaramakrishnan, Krishnan, & Mukhopadhyay, 2003). Agrawal, Khatri, & Srinivasan (2012) performed content analysis from 25 real-time case studies from the Indian software industry to identify significant human resource management challenges of software industries. The authors found that high IT employee turnover rate is the most recurring problem stated in 9 of the 25 case studies analyzed. The authors report that monotony of work, lack of growth and learning opportunities, high work pressure are some of the reasons that contribute to the high turnover rate in software organizations. The high turnover rate leads to loss of domain knowledge competencies resulting in project delays, and, hiring new professionals who are not ready to take up the project responsibilities, and also the loss of customer satisfaction. Casey, Deshpande, & Richardson (2010) performed telephonic interviews with five top management professional of the IT industry and identified that turnover rate is a serious problem in offshored and outsourced software projects as there is a high demand for technically competent staff in the offshore locations. The authors state that losing key

professional at critical phases of the projects can be detrimental to the project success. Although prior studies have identified that turnover is a major problem impeding the software development and qualitatively investigated about the reasons for tunover rates, empirical analysis about IT employee turnover rate is very rare. High employee turnover is a serious problem because frequently the software professional have to be shuffled impacting the quality and project schedules (Gopal & Koka, 2012; Lacity & Hirschheim, 1993). Therefore, investigating about the high turnover rates is a promising area of research given the serious implications it has on the software development outcomes (Kong, Chadee, & Raman, 2013). Specifically, the impact of turnover rate in CMM level projects and moderation effects, in particular, has not been investigated to date.

Team Work Experience

In the software industry, professional experience is often used to assess a person's technical capability and domain knowledge (Boehm, 1987; Faraj & Sproull, 2000; FP Jr, 1987). An experienced team will be able to foresee risks, manage budget and schedules (Barki, Rivard, & Talbot, 1993; Faraj & Sproull, 2000). Software development teams learn to deal with unique challenges and coping strategies from past experiences (Lee et al., 2006). Beaver & Schiavone (2006) proved that the development teams experience has significant positive impact on the software quality. Previous IS research has identified team development experience as one on the leading risk factors that influence the software project outcomes (Hu et al., 2013). However, there is also another side of the coin that having the right people with right attitude to learn and collaborate with others rather than experienced people ensures the success of projects (Lindvall et al., 2002). Lindvall et al. (2002) state that the percentage of competent and experienced staff can be as low as 10 % if the team members can coordinate and mentor each other. However, the authors state that the scenario could be different with other project outcomes. Easton & Rosenzweig (2012) analyzed the effects of team experience of six sigma projects on the likelihood of software project performance using four different indicators: individual experience, organizational experience,

team leader experience, and team familiarity. The authors found conflicting results than the prior literature that although team leader experience and corporate experience had significant positive impact on the project success, individual experience and team familiarity did not explain the project success. The authors suggest that well developed and structured problem-solving process reduce the need for the highly experienced development team in six sigma organizations. Clarke & O'Connor (2012) developed a framework for making decisions about implementing software development process with situational drivers of software development projects. They state that situational drivers such as task complexity, team experience, volatility of requirements, and culture of development team drive the process decisions.

Team Functional Heterogeneity

Heterogeneity means 'differences'. Embracing and managing differences in organizations is always a challenge. Team heterogeneity is the degree to which members of a team are distributed among different cultures, nations, functional expertise, and gender, age (Maloney & Zellmer-Bruhn, 2006). Software development is usually done in teams, which may have functionally diverse members in composition to integrate knowledge from various stakeholders like users, project manager's developers, clients etc. (Gopal & Gosain, 2010). Team heterogeneity is expected to have a powerful influence on the project outcomes, since it affects the knowledge and skill applied to the team based task (Somech & Drach-Zahavy, 2013). Among the various dimensions of heterogeneity, team functional heterogeneity is an indicator of team composition which is task related. Software organizations form functionally diverse teams depending on the requirements of the tasks which require people having knowledge in different job functionalities (Magjuka & Baldwin, 1991). Previous literature suggests that team heterogeneity has both positive and negative consequences. Magjuka & Baldwin (1991) suggest that heterogeneity in teams is a valued team resource since the teams are exposed to different information variety and richness to solve the complex problems. Although the general conception is that heterogeneity enhances team performance, this has not been consistently found in previous research (Cronin & Weingart, 2007;

Hambrick, Cho, & Chen, 1996; Joshi & Roh, 2009). Team heterogeneity can pose challenges in software development teams since most of the projects are globally dispersed leading to communication problems, interpretation issues; power struggles indirectly affecting the team effectiveness and project (Maloney & Zellmer-Bruhn, 2006). According to (Milliken & Martins, 1996) increased heterogeneity results in decreased cooperation, coordination, and cohesion among team members reducing the team performance. Cronin & Weingart (2007) suggest that functional heterogeneity encourages representational gaps and inconsistencies between team members in cross functional product development teams making it difficult for team members to integrate one another's information. Representational gaps make coordination difficult by creating contradictions in how teammates believe the problem should be solved, leading them to take actions that contradict each other, leading to differences and conflict between the members and makes the team far away from success. In software development, cross-functional teams reside in different locations adding up demographic heterogeneity. When functional heterogeneity combines with demographic heterogeneity, the problem gets much severe as the differences in understanding might augment. Given the important effects of team heterogeneity, the moderation effects of team heterogeneity in CMM level software projects is investigated.

1.3.4 Project Performance in IS Development:

Software development firms always face pressures to improve project performance measures such as productivity, time to market, product quality and customer satisfaction, etc. (Deephouse, Mukhopadhyay, Goldenson, & Kellner, 1995; Han, 2014). Project performance is the subject of prime importance for all the project stakeholders. The project performance criteria are multi-dimensional and are used to measure the success or failure of a project. These metrics differ from project to project based on the perceptions and related importance of success dimensions depending on the organization, nationality, project type, and contract type, etc. (P. Lu, Song, & Song, 2012; Söderland, Geraldi, Müller, & Jugdev, 2012). Sometimes software projects are expected to achieve all round performance in all performance aspects while others require

performance in specific performance areas (Han, 2014). An important step in improvising the project performance and deliver quality software is to measure the project outcomes, identify the loopholes, to rectify the problems in the future. Measuring and analyzing the project performance metrics helps control, evaluate, manage and influence the development process (Scacchi & Hurley, 1995). Project performance can be measured using two key aspects: process performance and product performance (Nidumolu, 1995). While the process performance, describes the performance of the software development process (time to market, productivity); product performance describes the performance of the system that was delivered to users (quality, customer satisfaction, etc.). Generally, software projects have multiple outcomes (Langer, Slaughter, & Mukhopadhyay, 2014); hence, several researchers have measured project performance using various metrics such as costs, elapsed time, productivity, and quality, etc. in the previous studies. For instance, Langer et al. (2014) used cost performance (actual expenses incurred in projects) and client satisfaction (actual rating for the customer satisfaction of a project) to measure project performance. Han (2014) used four measures to estimate the project performance: quality, time, customer satisfaction and budget to determine how the different categories of risks affect the project performance measures. In a recent study conducted by Athavale & Balaraman (2013), the authors used project performance measures such as quality and quantity of work completed in a unit time (productivity) to develop a human behavioral modeling simulation to demonstrate how human aspects affect task performance in a software project.

However, in this study, the process performance metrics: software productivity and elapsed time are used, which measure the efficiency of the CMM process under which the software projects are developed. Productivity helps reduce the cost of the software project and improves developer competitiveness and elapsed time denoted the quickness in delivering the software projects (Deephouse et al., 1995). Given the importance of process performance metrics, availability of data, and the focus on CMM level software projects, project performance is measured using the efficiency measures such as project elapsed time and productivity.

1.3.5 Research Gap

Although team characteristics are deemed to be important by prior researchers, empirical investigations are very few (Hoegl & Parboteeah, 2007). Moreover, their role in the CMM/CMMI projects has been under-explored. Therefore, this study tests the impact of team characteristics on the project performance to understand the importance of human factors in software development as shown in Figure 1.2.

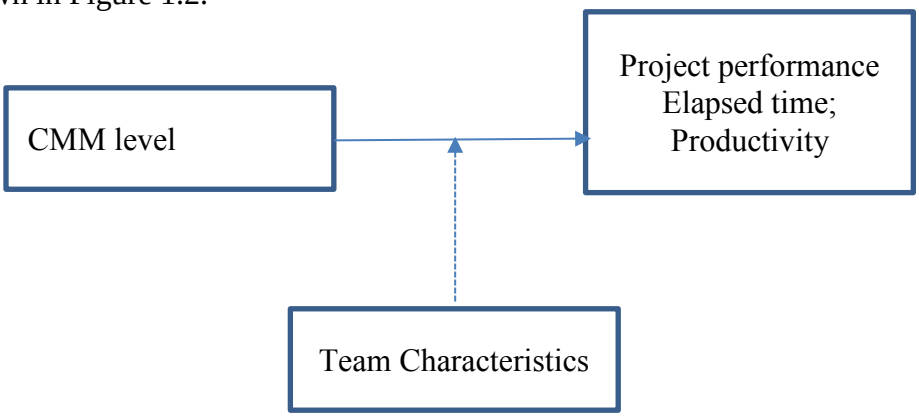


Figure 1.2: Effect of Team Characteristics on CMM Level and Project Performance

Apart from having a comprehensive process like CMM in place, having motivated, qualified and quality teams is also crucial for a project. In this study, the team characteristics such as team turnover, team functional heterogeneity and teamwork experience are analyzed. The main contribution of the paper is developing empirical models and evaluating the role of team characteristics on project performance in CMM level software development projects. In doing so, the following research questions are answered.

Research Questions

Research Question 1: Do team characteristics moderate the relationship between CMM/CMMI and project performance?

Research Question 2: What is the nature of moderation effect?

1.4 Hypotheses

As shown in Figure 1.2 the study leverages on the premise that implementing CMM process maturity in software development results in positive outcomes. Implementing CMM improves the productivity (i.e., the project delivery rate) of software projects and reduces the elapsed time. However, these positive effects can be modified depending on the team attributes such as team turnover (attrition within a team), functional heterogeneity, and teamwork experience. In this regard, six hypotheses for the study are developed.

1.4.1 Team Turnover

Turnover (voluntary and involuntary attrition) in the software industry is a challenging problem, and it is to be kept as much lower as possible to maintain the momentum and motivation of a team (Ebert, Murthy, & Jha, 2008). Houston et al. (2001) state that the unplanned departures of team members in a project before the agreed time can cause havoc to the project schedules since the replacement personnel has to catch up quickly with the project nuances that is not yet tested and complete (Reel, 1999). The turnover rate in a team can be a serious issue since new people have to be brought in and trained (Rajkumar and Mani 2001) which can impact the elapsed time of a project. Houston, Mackulak, & Collofello (2001) tested the effect of low morale on productivity and found that high turnover and schedule pressure moderate the confidence level thus, producing effects on productivity. Athreye, 2004; Korrapati & Eedara (2010) stated that in 1999, the software industry had responded to the problem of turnover by increasing the salaries of the professionals, this has, in turn, increased the productivity. Korrapati & Eedara (2010) noted that employee turnover could be a challenge to the IT companies that work towards meeting software project success regarding completing projects on time, within budget with utmost quality. Low absenteeism and low turnover were indirect benefits for Raytheon, a diversified, international, technology-based company when it achieved higher productivity upon evolving from level 1 to level 2, and levels 3 of the Software Engineering Institute's Capability Maturity Model (Dion, 1993; Ramanujan & Kesh, 2004). Therefore, high turnover can adversely impact the project

outcomes even though the projects implement best methodologies or process maturities since high turnover rate leads to the loss of knowledge and competencies in project teams (N. M. Agrawal et al., 2012) adversely affecting project delivery dates and productivity. In this study, both voluntary and involuntary departures due to turnover in a team are considered and it is termed as team turnover. Thus, the following hypotheses are proposed.

H1a: Team turnover and CMM level of a software project interact negatively such that the positive effect of CMM on the productivity is less strong as turnover increases.

H1b: Team turnover and CMM level of a software project interact positively such that the negative effect of CMM on the project elapsed time is less strong as turnover increases.

1.4.2Team Functional Heterogeneity

In this study, team heterogeneity is measured as the existence individuals belonging to different functional levels in a team. More number of functional levels indicate higher heterogeneity in the team. Prior studies about the relationship between heterogeneity and project performance have yielded mixed findings. Few researchers observed that heterogeneity in teams gave positive outcomes than homogeneity in teams (Nonaka & Takeuchi, 1995). Other studies concluded that homogeneity in teams could eliminate communication issues and conflicts that diverse teams usually encounter (Guzzo & Dickson, 1996). Gorla & Lam (2004) testified that a primary factor contributing to poor performance is software project team composition. Xia & Lee (2004) stated that project managers must develop strong relationships with top management and end users alike. Team heterogeneity is shown to have a positive impact on the team performance (Gorla & Lam, 2004). Involvement of different functional groups in a team leads to functional heterogeneity, and teams with functional heterogeneity outperform teams that lack such heterogeneity (Thomas-Hunt, Ogden, & Neale, 2003). Bhadury, Mighty, & Damar (2000) suggested a network flow approach to maximize workforce heterogeneity in project teams. Murray (1989) used social integration and communication patterns to predict the form of the relationship between team heterogeneity and organizational performance. The author argued that

team heterogeneity may lower performance in stable environments because the team would be less cohesive and require more formal communication. Through its effects on the workforce level, a project's schedule also affects productivity. For example, a higher workforce level means more communication and training overhead, affecting productivity negatively (Abdel-Hamid & Madnick, 1989). However, most of the studies have focused on the top management team heterogeneity but not at the development team level. In this study, the functional teams include software developers, support groups, users, testing groups and other project stakeholders. Therefore, the following two hypotheses are proposed.

H2a: Team functional heterogeneity and CMM level of a software project interact negatively such that the positive effect of CMM on the productivity is less strong as team functional heterogeneity increases.

H2b: Team functional heterogeneity and CMM level of a software project interact positively such that the negative effect of CMM on the project elapsed time is less strong as team functional heterogeneity increases.

1.4.3 Teamwork Experience

Understanding the role of work experience in software development is essential because experienced workforce constitutes the largest component of software project costs is the team (Fong Boh, Slaughter, & Espinosa, 2007). An individual gains knowledge through doing something thus gaining the necessary experience to identify process improvements (Syverson, 2011). Hence, work experience is an important factor for better project outcomes. According to Bloch et al. (2012), the right team would understand both business and technical concerns, which is why companies must assign a few high-performing and experienced experts for the length of the program. Michel & Hambrick (1992) used the concept of social integration to explain links between average team tenure and diversification strategy and performance. Cataldo, Herbsleb, & Carley (2008) noted that experienced software development personnel in different dimensions like tools, domain area, and programming can be more productive and can accelerate the development

time for a project. High management and staff experience contributed 120% to productivity while efficient methods/process contributes only 35% (Adolph et al., 2012). Fong Boh et al. (2007) analyzed whether individuals, groups, and organizational units learn from experience in software development and further examined if experience improves productivity. They found that specialization and diverse experience enhances productivity. Understanding the importance of work experience in software development projects the following hypotheses are proposed,

H3a: Team Work experience and CMM level of a software project interact positively such that the positive effect of CMM on the productivity is stronger as team work experience increases.

H3b: Team Work experience and CMM level of a software project interact negatively such that the negative effect of CMM on the project elapsed time is stronger as team work experience increases.

Summary of hypotheses is shown in Table 1.1

TABLE 1.1: SUMMARY OF HYPOTHESES

Hypotheses for Productivity	
H1a	Team turnover and CMM level of a software project interact negatively such that the positive effect of CMM on the productivity is less strong as turnover rate increases.
H2a	Team functional heterogeneity and CMM level of a software project interact negatively such that the positive effect of CMM on the productivity is less strong as team functional heterogeneity increases.
H3a	Work experience and CMM level of a software project interact positively such that the positive effect of CMM on the productivity is stronger as team work experience increases.
Hypotheses for project elapsed time	
H1b	Team turnover and CMM level of a software project interact positively such that the negative effect of CMM on the project elapsed time is less strong as turnover rate increases.
H2b	Team functional heterogeneity and CMM level of a software project interact positively such that the negative effect of CMM on the project elapsed time is less strong as team functional heterogeneity increases.
H3b	Work experience and CMM level of a software project interact negatively such that the negative effect of CMM on the project elapsed time is stronger as team work experience increases.

1.5 Research Model

In this section, the research model used for the study is presented in Figure 1.3. The central idea of the study design is that the maturity of a project has a positive impact on the project performance. It is expected that this relation will be moderated by the team characteristics of software project teams. Hence, the research model explores the three dimensions of team characteristics: team work experience, team turnover rate, and team functional heterogeneity, to test the moderation effects of team characteristics on the relationship between the maturity of projects and project performance measures. Project performance in software projects can be measured using various measures, yet, the most important measures such as productivity and the project elapsed time are used for the study.

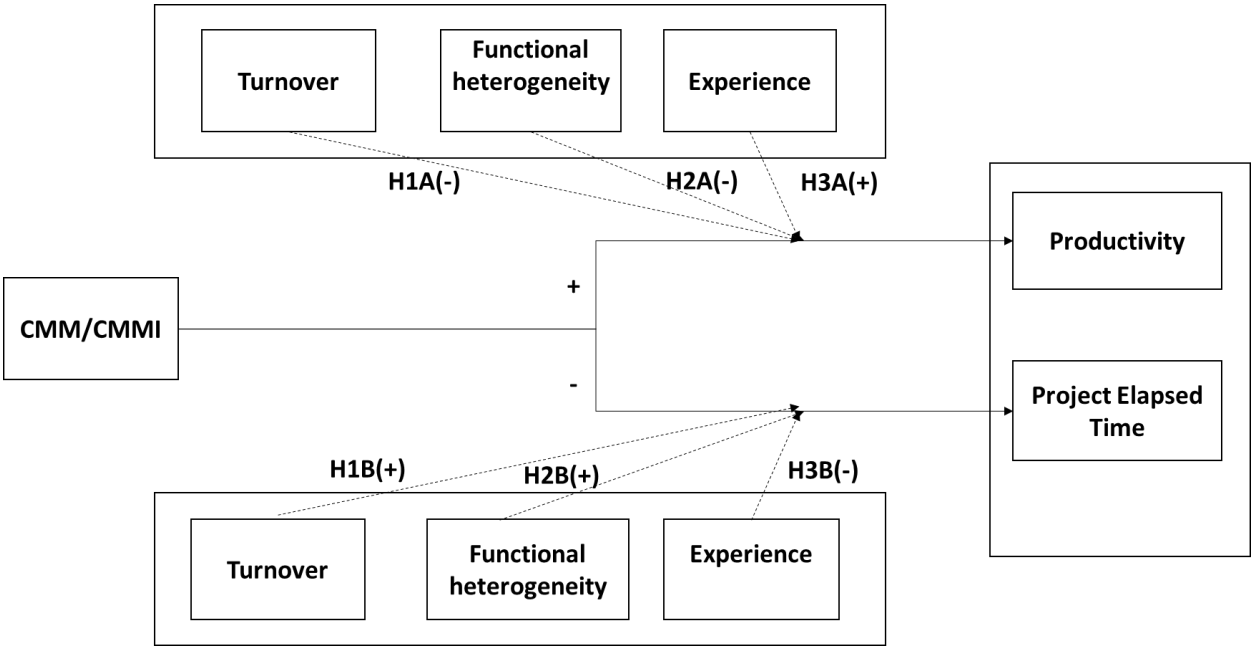


Figure 1.3: Research Model for the Study

1.6 DATA

The data used for this study is obtained from the International Software Benchmarking Standard Group (ISBSG) Release 12 repository containing information about software development and enhancement projects representing the top 25 % of the industry. The software

projects vary across various business areas from 20 different countries. This data has been reported voluntarily by the various software organizations and industry leaders. It has also been validated based on ISBSG quality guidelines and was used in some of the previous studies (Bagchi et al. 2015; Jiang et al. 2007; Oligny et al. 2000). ISBSG is a reliable database given the number of research articles published in software development research (Fernández-Diego & González-Ladrón-de-Guevara, 2014). However, data for all the areas was not reported for all the projects. Since the study focusses on the maturity level of the software projects, 279 CMM and CMMI level projects, implemented between 1999 and 2011 are identified. Distribution of the projects based on CMM level and the industry sector is shown in Table 1.2 and Table 1.3.

TABLE 1.2: DISTRIBUTION OF PROJECTS BASED ON THE CMM LEVEL

CMM/CMMI	No of projects
level 1	6
level 2	136
level 3	60
level 4	9
level 5	68

TABLE 1.3: DISTRIBUTION OF PROJECTS BASED ON THE INDUSTRY SECTOR

Industry Sector	**Number of projects**
Banking	10
Communication	35
Construction	1
Electronics & Computers	5
Energy Sources	1
Financial	7
Government	104
Insurance	17
Manufacturing	43
Medical and Health Care	8
Professional Services	5
Service Industry	36
Wholesale and retail	6
Other	1

Team characteristics of these CMM and CMMI level projects such as the number of years of work experience, the number of functional levels in a team (functional heterogeneity) and the number of changes in the team members during the tenure of the project (team turnover) are used for the analysis.

1.6.1 Measurement of the Constructs

Dependent Variable Constructs

The dependent variables used in the model are the project performance measures: productivity and project elapsed time.

Productivity

Productivity is an important measure for every business especially software development since it helps to 1) determine if one methodology produces better results or not 2) make decisions about the most cost-effective tools and techniques 3) compare a development team to industry competitors 4) make decisions optimum team size (http://isbsg.org). Productivity is the cost to produce a batch of items (the input measure) divided by the number of items produced (output measure). According to ISBSG, productivity is the project delivery rate, and it is measured using effort required to build the system divided by the size (measured in function points) of the software. In this study, the inverse of the productivity as calculated by the ISBSG is calculated and is called as project delivery rate. Therefore, productivity according to this study is the number of function points delivered per work effort. It helps determine the speed with which the project has to be delivered. A higher value for the project delivery rate is desired to have better productivity since it shows that more number of function points are delivered per unit of work effort. Therefore, the productivity is a formative construct measured by the project delivery rate.

Project Elapsed Time

The other dependent variable used in the research model is the project elapsed time. According to the ISBSG, this is the total elapsed time taken for the completion of the project in months. Project managers often face the task of providing information about the likely size, cost,

and duration of a proposed project to assist with a business decision about the project's feasibility. Project managers can estimate the duration of a project using the software size, the software development life cycle (SDLC) phase profile, the overlap between SDLC phases and activities in software development, staff availability and other factors by converting them into the number of calendar months. Having an assessment of the project elapsed time helps answer questions such as 1) Is it practical or possible, to deliver a software project with an estimated project size within the target time frame? 2) If I have a team of 'n' people, what is the likely elapsed time for the project? Therefore, in the model, this measure is used to analyze the impact of team characteristics to provide a better understanding of the planning of the software development activities. Hence, the project elapsed time is a formative construct measured by the number of months taken to deliver the project.

Independent Variable Constructs

The independent variable constructs used in this study are team work experience, team functional heterogeneity, and team turnover.

CMM level

CMM level represents the CMM level of the organization under which the software projects were developed.

Team Turnover

Team turnover represents the number of unexpected changes in the team members that occurred in a software development team during the tenure of the software project. turnover in a software team can happen due to the decision taken by the top management or due to the voluntary decision of the team members and could range from project managers to project personnel. The study considers turnover occurring because of any abnormal replacements due to the illness of the team members, replacements due to nonperformance, voluntary resignations. The team turnover is a formative construct measured using two attributes: number of changes in the project manager and changes in the project personnel during the elapsed time of a particular project.

Team Heterogeneity

Team heterogeneity is the functional heterogeneity existing in the software team. Software development requires personnel from different functional diversities or levels to come together and work as a team depending on the personnel from. Team heterogeneity construct in this study is measured using four resource levels. It is measured using a single item attribute which includes all the people whose time is included in the work effort for a project. Each of the four resource levels represents the following. Level l includes all the software development team personnel (e.g., project team, project management, project administration). Level 2 includes development team support personnel (e.g., database management, data administration, quality assurance, data security, standards support, audit & control, technical support. Level 3 includes computer operations involvement (e.g., software support, hardware support, information center support, computer operators, network administration). Level 4 includes end users or clients (e.g., user liaisons, user training time, application users and/or clients). A number in this field indicates that all effort at this and preceding levels are included in the effort fields, where a high value indicates high heterogeneity. For instance, a value of 1 in team heterogeneity measure indicates that only level 1 personnel are involved in the software project, and a value of 4 shows that all the four levels (from level 1 to level 4 staff are involved in the software project.

Work Experience

Work experience is an important characteristic that helps assess the capabilities of the development team members. It is measured by the number of years the team members worked in the software development projects. The software development teams primarily comprise of project managers, developers, business analysts, etc. Hence, in this study the work experience is a formative construct measured using two variables: Total work experience of the information technology team members and the overall professional experience of the business analyst team members. Although project manager experience is also a necessary measure, due to unavailability of data, it was not included for measuring the construct.

1.7 Research Method

Structural Equation Model (SEM) is employed to test the hypotheses about the moderation effects of the team characteristics on project elapsed time and productivity. For this, WARPPLS 5 (Kock, 2015) software is used to test the model. Structural equation models go beyond ordinary regression models to incorporate multiple independent and dependent variables as well as hypothetical latent constructs that clusters of observed variables might represent (Savalei & Bentler, 2010). The non-covariance based PLS-SEM is chosen over the CB-SEM because PLS-SEM can work efficiently with a much wider range of sample sizes and increased model complexity. With its less restrictive assumptions about the data, it can address a broader range of problems than CB-SEM (Hair, Ringle, & Sarstedt, 2011; M. H. N. Nasir & Sahibuddin, 2011). Moreover, PLS-SEM is suited when the study is about prediction rather than confirmation of the structural relationship. It allows for simultaneous assessment of multiple dependent and independent constructs, including multi-step paths (Gebauer, Kline, & He, 2011; Gefen, Straub, & Boudreau, 2000).

In this study, the moderation effects of team characteristics on the relation between the CMM level of software projects and the performance metrics such as project elapsed time and productivity are tested. Here the dependent variables of the study are the project performance measures productivity and the elapsed time. Based on the previous studies it is identified that implementing the CMM in software organizations improves the productivity. Through this study, the moderation effects of team professional experience, team functional heterogeneity, and team turnover are tested.

The issues of moderators have long been existing in psychology literature mainly simulated by (Baron & Kenny, 1986). The moderators are variables that specify for whom or under what conditions a particular relation works. A moderator effect exists when the third variable (qualitative or quantitative that affects the direction and strength: either reduces or increases or reverses the relation between an independent or predictor variable and a dependent or criterion variable. The moderator hypothesis is supported if the interaction is significant. There may also be

significant main effects for the predictor and the moderator, but these are not directly relevant conceptually to testing the moderator hypothesis. Testing the moderation effects help clarify to investigators the best choice of inclusion and exclusion criteria or the best option of stratification to maximize power in subsequent relation. Moderators may identify subpopulations of samples with possibly different causal mechanisms where the relation between variables might reverse due to the existence of moderators. Thus, moderators may also provide unique new and valuable information to guide team reconstruction and decision making. According to David and Kenny (http://davidakenny.net/cm/moderation.htm) the moderation effects can be positive or negative, the effect is positive if the effect of an independent variable on the predictor variable increases as the value of the moderator variable increases. The effect is negative if the effect of independent variable on the predictor variable decreases as the value of the moderator variable increases. The positive and negative effects can be tested using simple slope analysis suggested by (Aiken, West, & Reno, 1991). In a simple slope analysis, the linear relation between the predictor and the independent variable is plotted both for high and low values of the moderator variable. A steep slope of the relationship indicates a stronger relationship between the predictor and the independent variable. Thus, the positive or negative moderation effects of the work experience, functional heterogeneity, and the turnover are determined to identify the best strategies to maximize the positive relation between CMM and project performance characteristics elapsed time and productivity.

The research model of the study is tested in two stages. In the first stage, the model is tested only for the main effects (Model 1). In the second stage, the model is tested for both the main and the moderation effects (Model 2). The analysis of the results from the SEM analysis is performed in two stages. In the first stage, the overall fit of the model was tested using the model fit indices suggested by (Kock, 2015) and the measurement model is validated using the convergent and discriminant validity. In stage two, the structural model is analyzed to test the hypotheses and the path coefficients for the relationships between the exogenous (path arrows pointing outwards and

none leading to it) and the endogenous variables (variables have at least one path leading to it or dependent variables).

1.8 Results

The model fit indices from the PLS analysis for the Model1 and Model 2 are shown in Table 1.4.

TABLE 1.4: MODEL FIT INDICES

Model fit indices	Model 1	Model 2	Recommended value
Average path coefficient (APC)	0.162;P=0.001	0.134, P=0.006	p<0.05
Average R-squared (ARS)	0.204, P<0.001	0.243, P<0.001	p<0.05
Average adjusted R-squared (AARS)	0.193, P<0.001	0.223, P<0.001	p<0.05
Average block VIF (AVIF)	1.259	1.449	acceptable if <= 5, ideally <= 3.3
Average full collinearity VIF (AFVIF)	1.277	1.58	acceptable if <= 5, ideally <= 3.3
Tenenhaus GoF (GoF)	0.438	0.474	small >= 0.1, medium >= 0.25, large >= 0.36
Sympson's paradox ratio (SPR)	1	0.929	acceptable if >= 0.7, ideally = 1
R-squared contribution ratio (RSCR)	1	0.984	acceptable if >= 0.9, ideally = 1
Statistical suppression ratio (SSR)	1	0.929	acceptable if >= 0.7
Nonlinear bivariate causality direction ratio (NLBCDR)	1	0.893	acceptable if >= 0.7

According to Kock (2015), it is recommended that the P-values for the Average path coefficient (APC), Average R-squared (ARS) and Average Adjusted R-squared (AARS) all be equal to or lower than 0.05; that is, significant at the 0.05 level for model fit and quality. Both Model 1 and Model 2 satisfy this condition. It can also be observed that the average R-square and average adjusted R-square increase after moderation effects are added to the model. This shows that the model with the moderation effects (Model 2) (ARS=0.204; AARS=0.243) has a better fit

compared to the main effects model (Model 1) (ARS =0.193; AARS=0.223). The Tenenhaus GoF index is a measure of a model's explanatory power. For both the models, results show that the GOF value is greater than 0.5 indicating that they have high explanatory power and compared to the main effects model (Model 1), the moderation effects model (Model 2) have higher explanatory power. Thus, both the models are stable with the Model 2 explaining a greater amount of variance and with high explanatory power.

Further analysis of results is performed in two steps. Firstly, the measurement models are validated using convergent and discriminant validity. Secondly, the path coefficients of the structural model are analyzed to test the hypothesis.

1.8.1 Measurement Model

The convergent validity checks the pattern of loadings of the measurement items into the latent variables and discriminant validity measures the quality of the measurement instrument. Convergent validity was tested by performing the confirmatory factor analysis with the factor loadings and cross-loadings. Two criteria are recommended as the basis for concluding that a measurement model has acceptable convergent validity: the P values associated with the loadings be equal to or lower than 0.05; and the loadings be equal to or greater than 0.5 (Hair, Anderson, Tatham, & William, 1998; Kock, 2015). Results from the confirmatory factor analysis show that all the observed variables load well into their latent constructs without any significant cross-loadings as shown in Table 1.5.

Discriminant validity was tested by comparing the average variance extracted to the correlations among the latent variables. For discriminant validity, the square root of average variance extracted should be greater than the any of the correlation involving the latent variable (Fornell & Larcker, 1981; Nunnally & Bernstein, 1994). Correlation between latent constructs and average variance extracted (AVE) are shown in the diagonal shown in Table 1.6. The table indicates that the AVE is greater than the correlations between the constructs thus showing evidence of discriminant validity.

TABLE 1.5: NORMALIZED PATTERN LOADINGS AND CROSS LOADINGS FOR MODEL2

Variables	P value	Productivity	Elapsed time	Team diversity	Experience	Attrition	cmm
Productivity	<0.001	**1**	0	1	0	0	0
Project Elapsed time	<0.001	0	**1**	0	1	0	0
No. of Resource level	<0.001	0	0	**1**	0	0	0
IT developer experience	<0.001	-0.083	0.009	-0.026	**0.965**	0.135	-0.026
Analyst experience	<0.001	0.083	-0.008	0.024	**0.967**	-0.127	-0.024
Personnel changes	<0.001	-0.005	0.031	0.005	-0.01	**0.981**	-0.093
Manager changes	<0.001	0.005	-0.031	-0.005	0.01	**0.982**	0.092
CMM level	<0.001	0	0	0	0	0	**1**
For convergent validity, P values associated with the loadings be equal to or lower than 0.05; and that the loadings be equal to or greater than 0.5							

TABLE 1.6: INTER CONSTRUCT CORRELATIONS AND DISCRIMINANT VALIDITY FOR MODEL 2.

Constructs	Productivity	Elapsed time	Team heterogeneity	Experience	Team turnover	CMM/CMMI
Productivity	**1**					
Elapsed time	-0.16	**1**				
Team heterogeneity	-0.1	-0.07	**1**			
Experience	-0.27	0.14	0.01	**0.88**		
Team turnover	-0.22	0.24	0.04	0.57	**0.75**	
CMM/CMMI	0.28	-0.2	-0.15	-0.16	-0.04	**1**
Inter construct Correlations Table: Diagonal elements in the correlation of constructs matrix are the square root of the average variance extracted. For adequate discriminant validity, diagonal elements (composite reliabilities) should be greater than corresponding off-diagonal elements.						

Reliability of the constructs is measured using the composite reliabilities of the latent variables. According to Fornell & Larcker (1981) and Nunnally & Bernstein (1994), the composite reliabilities of latent variables have to be greater than 0.7 for construct reliability. The Value Inflation Factors for both model 1 and model 2 were also tested to check for multicollinearity issues between the latent variables which indicate that two latent variables measure the same thing (Hair et al., 1998). Results show that there are no multicollinearity issues between the latent variables.

1.8.2 Structural Model

The structural model for the Model 1 tests the main effects of the team characteristics on the software project outcomes: productivity and project elapsed time whereas the Model2 examines both the main and the moderator effects. The change in R-square between the Model1 and Model2 shows the effect size of the moderators in the tested models. PLS-SEM results show that the total variance explained by Model 1 for productivity (PDR) as the Dependent variable (DV) is 16 % and for project elapsed time (PET) as the DV is 24 %, whereas, for model2, the total variance explained for productivity is 16 % and for project elapsed time is 32 %. The change in R-square between Model1 and Model2 for project elapsed time as DV is 9 percent and for the productivity as DV, the change is comparatively less but could be existing. It is evident from the model fit indices of the PLS analysis that there is an increase in the explained variance when the moderation effects are added to the model in addition to the main effects. This increase shows that team characteristics significantly moderate the relation between the CMM level and the project outcomes, productivity and the project elapsed time. Therefore, structural model and the moderation effects are analyzed using the results from the model 2. Further the moderation effects were also analyzed using simple slope analysis suggested by Aiken and West (1991) and Jeremy

and Dawson (2014). Table 1.7 and Table 1.8 show the path coefficients of the structural models. Figure 1.4 displays the research model with the results.

TABLE 1.7: RESULTS FROM PLS ANALYSIS FOR PRODUCTIVITY AS THE DEPENDENT VARIABLE

Path	Model 1	Model 2
CMM→PDR	0.21*	0.18*
Experience→ PDR	-0.21*	-0.21*
Team functional heterogeneity→ PDR	-0.06 ns	-0.1**
Team turnover →PDR	-0.09**	-0.03 ns
CMM*experience→ PDR		0.11**
CMM*functional heterogeneity → PDR		-0.1**
CMM* turnover → PDR		0.09***
R squared	**0.16**	**0.16**
CMM = Maturity of software project process; PDR = Productivity; * = significant at < 0.001; ** = significant at < 0.05; *** = significant at <0.1; ns = non-significant; ns = non-significant		

TABLE 1.8: RESULTS FROM PLS ANALYSIS FOR PROJECT ELAPSED TIME (DEPENDENT VARIABLE)

Path	Model 1	Model 2
CMM→PET	-0.37*	-0.31*
Experience→ PET	0.03 ns	0.06 ns
Team functional heterogeneity → PET	-0.08***	-0.11**
Team turnover →PET	0.26*	0.27*
CMM*experience→ PET		0.09**
CMM*functional heterogeneity → PET		0.22*
CMM* turnover → PET		-0.01 ns
R squared	**0.23**	**0.32**
CMM = Maturity of software project process; PET = Project Elapsed Time; * = significant at < 0.001; ** = significant at < 0.05; *** = significant at <0.1; ns = non-significant; ns = non-significant		

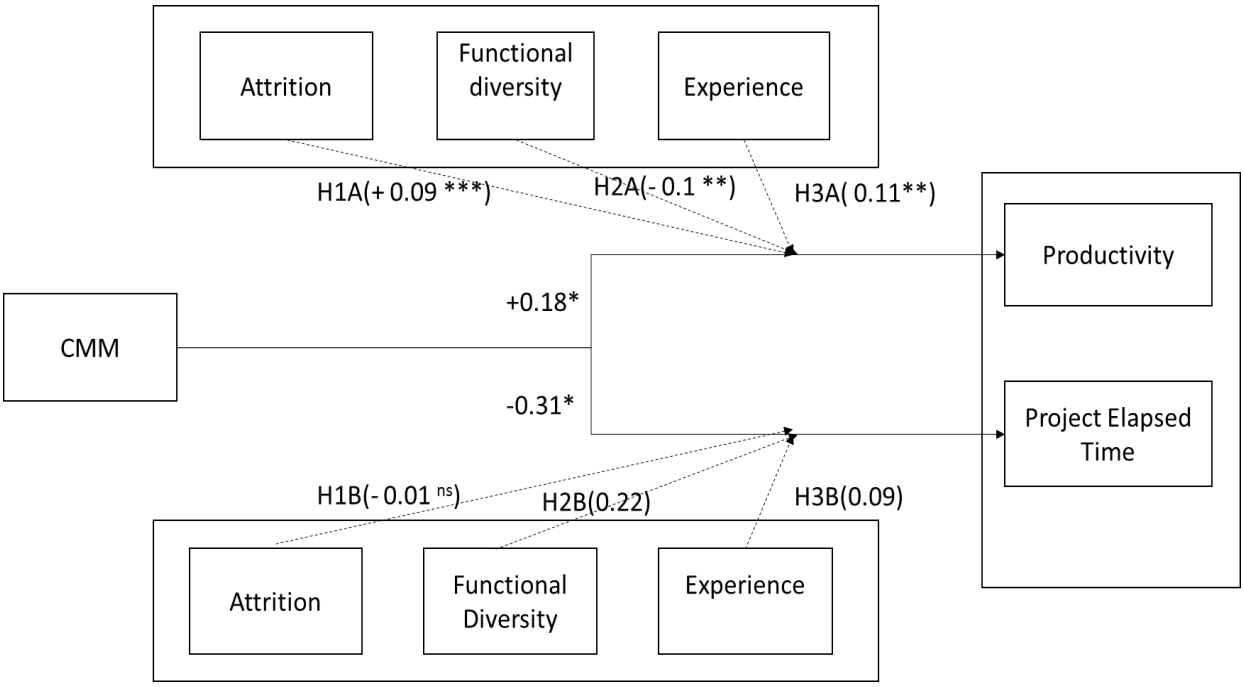


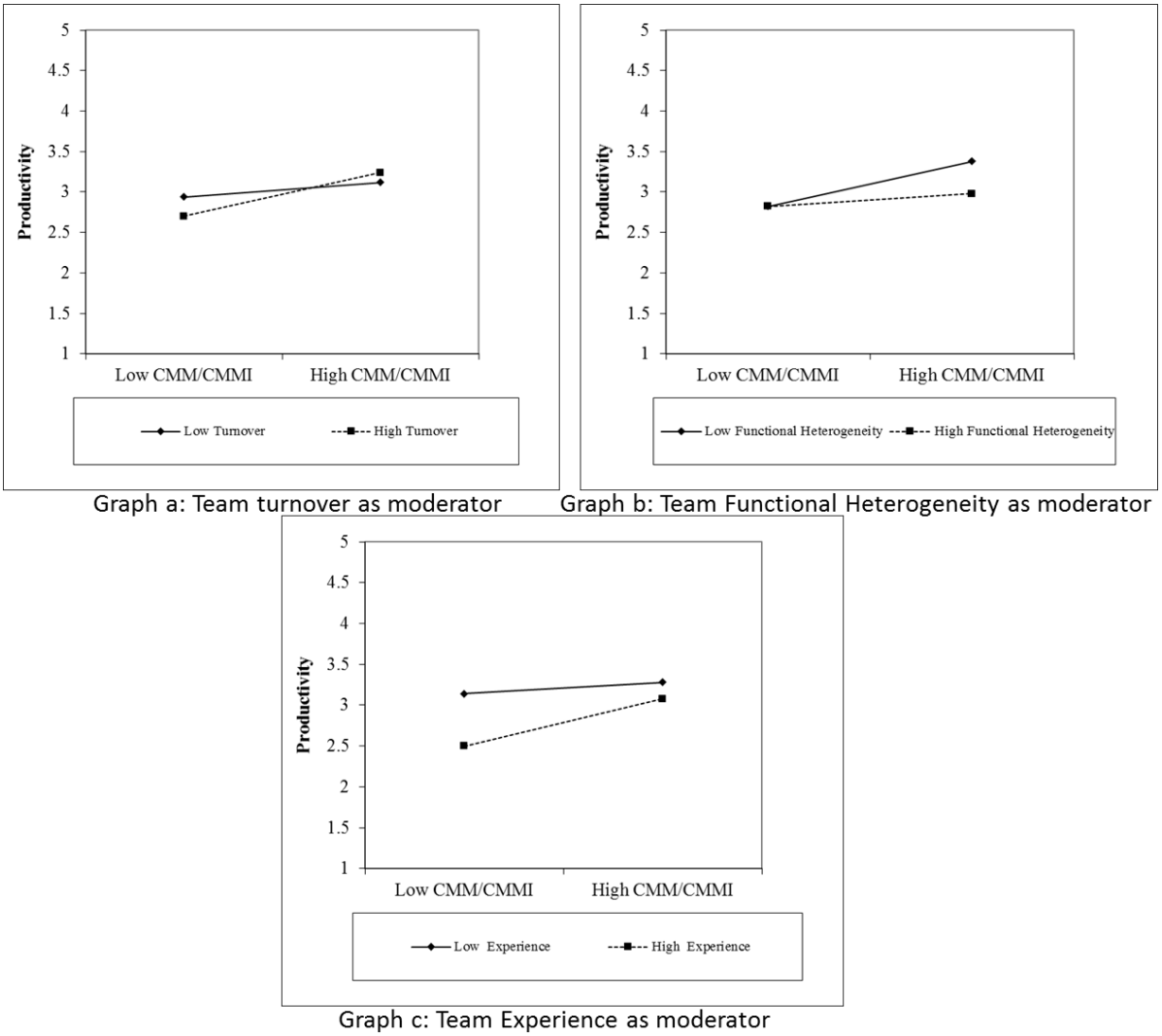
Figure 1.4: Results of the Research Model

Results for Productivity

Main effects (Model 1): It is observed from the main effects model (Model1) that, CMM level under which a software project is undertaken has a significant positive impact (β=0.21; p<0.001) on the productivity. The main effect of the team characteristics from the Model1 show that team turnover (β= -0.09; p=0.07) has a significant negative impact on the productivity. Work experience (β= -0.21; p<0.001) has a significant negative effect on productivity which is contradictory to the existing notion that having highly experienced people in a team always gives better results. However, team functional heterogeneity (β= -0.06; p=0.165) did not have any significant effect on the productivity.

Moderation effects (Model 2): The moderation effects are graphically displayed in Figure 1.5 (Graph a, Graph b, and Graph c) (Aiken, West, & Reno, 1991) to understand the nature of the

moderation effects of the team attributes used in the study. The dependent variable productivity is shown along the vertical axis, while the maturity level of the project is shown along the horizontal axis. After the moderation effects of the team characteristics were introduced along with the main effects, it is observed that all the three moderation effects were significant showing that team characteristics cast their interactions on the relation between CMM and the productivity.



Graph a: Team turnover as moderator

Graph b: Team Functional Heterogeneity as moderator

Graph c: Team Experience as moderator

Figure 1.5: Moderation Effects of Team Characteristics on Productivity in CMM/CMMI level software projects

Moderation effect results show that all the three team characteristics, team turnover (β= 0.09; p=0.08), team functional heterogeneity (β= -0.1; p=0.05), team experience (β= 0.11; p=0.04), have significant impact the relation between CMM and the productivity. Thus, stating that high or low functional heterogeneity, turnover and work experience of a software development project team have the capacity to strengthen or weaken the productivity in the CMM level projects. Results from the Model2 also suggest that team work experience and team functional heterogeneity partially moderate whereas, team turnover fully moderates the relation between CMM and productivity. This is evident because the main effect of team turnover became insignificant after the introduction of the moderation effect.

All the three graphs in Figure 1.5 (Graph a, Graph b, and Graph c) that explain the moderation effects of workforce characteristics emphasize that higher maturity levels of the software project result in higher productivity. However, the positive effect differs with the low and high values of the team characteristics which can be understood from the difference in the slope of the lines that represent high and low values of the team characteristics. Graph 1.5a shows the positive effect of CMM level of a software project on the productivity for low and high levels of team turnover. However, the positive effect is found to be stronger when the teams have a high turnover in a team. Thus, not supporting the hypothesis H1a that team turnover negatively moderates the effect of CMM on productivity. In Graph 1.5b, the effect of the CMM level of a project on the productivity of a software project for low and high team functional heterogeneity is shown. This graph shows that higher CMM levels of the software project result in increased productivity, yet the positive effect is different for high and low team functional diversity. The positive effect is stronger when the teams have less functional diversity. Thus, supporting hypothesis H2a that team functional heterogeneity negatively moderates the relation between

CMM and productivity. In Graph 1.5c, the effect of the CMM level of a software project on the productivity for low and high value of team work experience in a software project is projected. The graph shows that CMM level of the software project has a strong positive relationship with productivity, yet this relation is much stronger when the development team has high team work experience. Thus, supporting hypothesis H3a that team work experience positively moderates the relationship between CMM and the productivity. The summary of results of the hypotheses for productivity as dependent variable is shown in Table 1.9.

TABLE 1.9: SUMMARY OF HYPOTHESES FOR PRODUCTIVITY

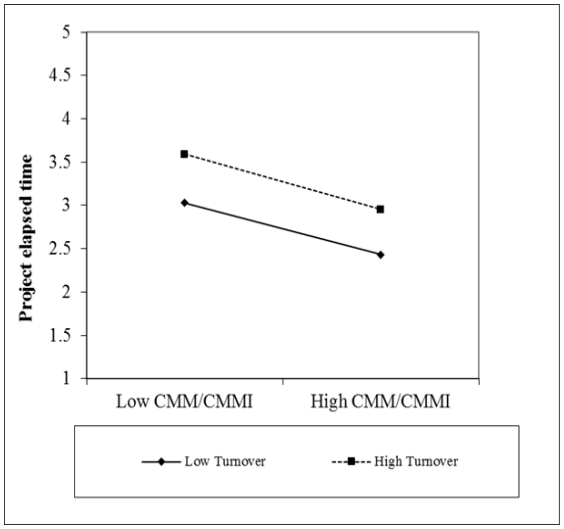
Hypotheses for productivity		Result
H1a	Team turnover and CMM level of a software project interact negatively such that the positive effect of CMM on the productivity is less strong as turnover rate increases.	Significant but not supported
H2a	Team functional heterogeneity and CMM level of a software project interact negatively such that the positive effect of CMM on the productivity is less strong as team functional heterogeneity increases.	Supported
H3a	Work experience and CMM level of a software project interact positively such that the positive effect of CMM on the productivity is stronger as team work experience increases.	Supported

Results for Project Elapsed Time:

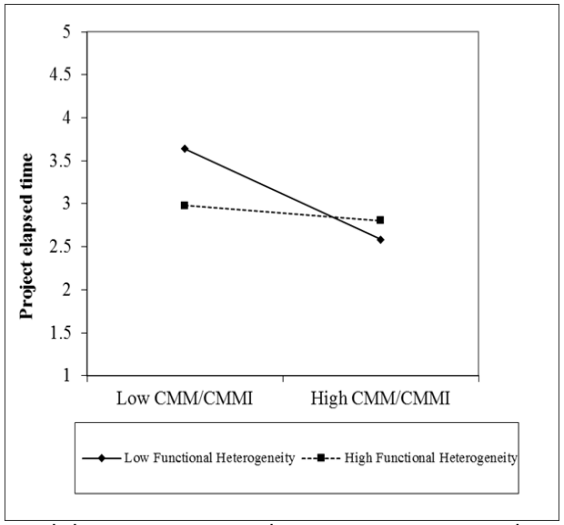
Main effects (Model 1): It is observed from the main effects model (Model1) that CMM level under which a software project is undertaken has a significant negative effect (β= -0.37; $p<0.001$) on the project elapsed time. This significant effect shows that implementing CMM levels in software organizations can help reduce the overall time of delivery of the projects. The main effects of the team characteristics from the Model1 show that team functional heterogeneity (β= -0.08; $p=0.094$) has a significant negative impact on the project elapsed time showing that projects involving team members from different functional background take less time to get completed. The team work experience (β= 0.03; $p<0.321$) did not have any significant impact on the project elapsed time. Team turnover (β= 0.26; $p<0.001$) has a significant positive effect on project elapsed

time showing evidence that teams having high turnover cannot give good performance since people leave organizations even before the project is completed.

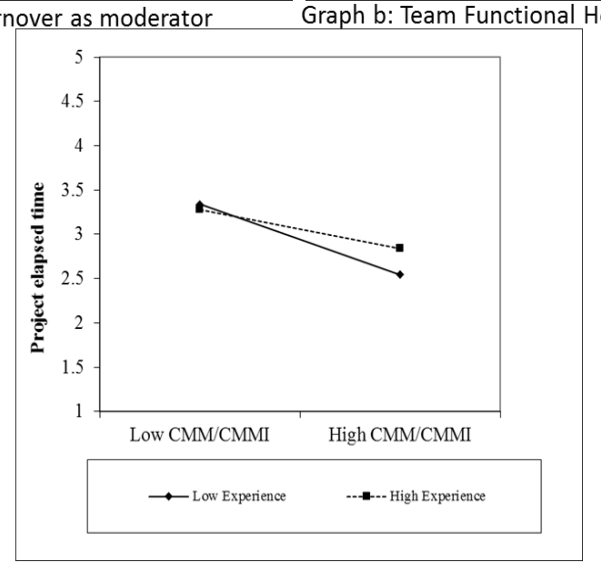
Moderation effects (Model 2): The moderation effects are graphically displayed in Figure 1.6 (Graph a, Graph b, and Graph c) (Aiken, West, & Reno, 1991) to understand the nature of the moderation effects of the team attributes used in the study.



Graph a: Team turnover as moderator



Graph b: Team Functional Heterogeneity as moderator



Graph c: Team Experience as moderator

Figure 1.6: Moderation Effects of Team Characteristics on Project Elapsed Time in CMM/CMMI and Elapsed Time

The dependent variable project elapsed time is shown along the vertical axis, while the maturity level of the project is shown along the horizontal axis. After the moderation effects of the team characteristics are introduced along with the main effects, it is found that out of the three team characteristics only team work experience ($\beta = 0.09$; $p<0.07$) and team functional heterogeneity ($\beta = 0.22$; $p<0.001$) have significant moderation effects on the project elapsed time. Thus, stating that high or low values of team functional heterogeneity and team work experience have the capacity to strengthen or weaken the project elapsed time in the CMM level projects. Team turnover ($\beta = -0.01$; $p<0.41$) did not have any significant moderation effect suggesting that high and low values of team turnover do not have any effect on the project elapsed time. Results from the Model2 also indicate that team functional heterogeneity partially moderates whereas team work experience fully moderates the relation between CMM and project elapsed time. This is evident because the main effect of team work experience became insignificant after the introduction of the moderation effect.

All the three graphs in Figure 1.6 (Graph a, Graph b, and Graph c) that explain the moderation effects of team characteristics emphasize that higher maturity levels of the software project help reduce project elapsed time, thus helping the project delivery in less time. This negative effect can be observed from the negative slope of the trend lines that depict the relation between the CMM and the project elapsed time. However, the results show that the negative effect differs with the low and high values of the team characteristics: team work experience and team functional heterogeneity which can be understood from the difference in the slope of the lines that represent high and low values of the team characteristics. In Graph a, the effect of the CMM level of a software project on the project elapsed time for low and high value of team turnover in a software project is projected. The graph shows that CMM level of the software project has a strong positive relationship with productivity, yet, this relation does not differ between teams having high and low turnover due to the insignificant results from the PLS analysis. Thus, not supporting hypothesis H1b. In Graph 1.6b the effect of the CMM level of a project on the project elapsed time of a software project for low and high values of team functional heterogeneity is shown. This graph

shows that higher CMM levels of the software project results in reduced elapsed time and the negative effect is different for high and low team functional heterogeneity. The negative effect is stronger when the teams have less functional heterogeneity. Thus, supporting hypothesis H2b. Graph 1.6c shows the negative effect of CMM level of a software project on the project elapsed time for low and high levels of team experience. However, the negative effect is found to be stronger when the teams have less experienced members in a team. Thus, not supporting the hypothesis H3b that work experience negatively moderates the effect of CMM level on project elapsed time. The summary of results of the hypotheses for project elapsed time as the dependent variable is shown in Table 1.10.

TABLE 1.10: RESULTS OF HYPOTHESES FOR PROJECT ELAPSED TIME

	Hypotheses for project elapsed time	**Result**
H1b	Team turnover and CMM level of a software project interact positively such that the negative effect of CMM on the project elapsed time is less strong as turnover rate increases.	Non-Significant
H2b	Team functional heterogeneity and CMM level of a software project interact positively such that the negative effect of CMM on the project elapsed time is less strong as team functional heterogeneity increases.	Supported
H3b	Work experience and CMM level of a software project interact negatively such that the negative effect of CMM on the project elapsed time is stronger as team work experience increases.	Significant but not supported

1.9 Discussion

Software process improvement initiatives have become integral components of software development organizations. Through this study, strong support and consensus is observed with previous studies (Diaz & King, 2002; Galin & Avrahami, 2006; Subramanian, Jiang, & Klein, 2007) that increase in the CMM level of an organization increases the success rate of software projects. Specifically, empirical support that CMM level can help increase the productivity and reduce the elapsed time of software development project is achieved. The effect of CMM level implementation can manifest in different forms in the software organizations. The Capability Maturity levels in software organizations ensure that any changes that are introduced in software

development process are implemented in an organized process and the amount of rework in organizations is reduced by identifying common processes. CMM also ensure that employees are adequately trained for the job and are well coordinated with their team members by streamlining the software development process. Thorough performance evaluations in CMM level organizations help them sustain the software development success rate. This success rate increases as the software organizations grow a level higher in the maturity levels.

In addition to finding empirical evidence about the benefits of implementing the CMM levels in software organizations, strong support was found that team factors play a significant role on the success or failure of software projects developed by the CMM level organizations. Most organizations ignore the effects of human factors and highlight the need for software development process initiatives (Baddoo & Hall, 2002; Ply, Moore, Williams, & Thatcher, 2012). However, this study strengthens the view of Laporte & Trudel (1998) that the software process improvement initiatives success depends more on the human aspects. This is supported by the significant moderation effects of the team characteristics on the CMM and the software project outcomes (productivity and elapsed time) relationship.

Moderation effect results about team turnover in software projects suggested contradictory explanations that teams with high team turnover have a stronger positive effect on the CMM productivity relationship whereas team turnover did not have a significant effect on the project elapsed time. The stronger positive effect for teams with high turnover can be explained because, turnover in this study not only considers replacements that occurred in a project due to the voluntary exit of employees due to resignations or transfers but also considers replacement due to illness, maternity leave, nonperformance. It could be that most projects had high turnover because employees were moved out of the projects due to nonperformance or under performance. While losing team members can disrupt the well-functioning of the teams. It can also lead to new opportunities by introducing new knowledge and also change existing routine practices in the team (Sarma & Herbsleb, 2008). Therefore, teams having high turnover could have a stronger positive effect on the CMM – productivity relationship since the non-performers were replaced with better

performing team members. Moreover the non-significant effect of team turnover on the CMM-project elapsed time relationship could be because, in CMM level organizations proper care is taken to have the development process standardized which can eliminate the hassles of project replacements to catch up with the work left by the prior employees (Lee, Espinosa, & DeLone, 2013). Therefore team replacements do not always lead to bad consequences as it can help teams to perform better by bringing in better performing employees.

It is observed from the results that teams with highly experienced members perform well in terms of productivity. This can be due to the fact that professional experience is often considered as a proxy for domain knowledge and the experienced professional already have a hands-on experience of working on similar projects (Espinosa, Slaughter, Kraut, & Herbsleb, 2007; Ramasubbu, Cataldo, Balan, & Herbsleb, 2011). This prior experience will help then to have a clear overview of the project which helps then to handle work efficiently. They can easily resolve any defects or bottlenecks in the architecture or design of the software project. Hence, the effort put by them in the project can help improve the productivity since they can be very effective in the effort that they put in the project that they work. Although the impact of work experience was as expected for productivity, the impact of work experience on project elapsed time was found to be contradicting the general expectation that it would reduce the elapsed time. It is found that projects having highly experienced team members have higher overall elapsed time of the project. This can be explained due to the fact that experienced members may be involved in many activities such as in project update sessions, resolving issue of other projects that they are involved in, training new developers etc. This leaves them with less time to work on a particular project and in turn increasing the elapsed time of the project. Although they may be effective in the time they work, they cannot control the other activities that act as hindrances which can ultimately extended timelines for the projects. Results about work experience suggest that software development teams should comprise of both experienced and less experienced developers to maintain good levels of productivity as well as elapsed time. Although experienced members can be beneficial to the software project, having too many highly experienced staff can lead to exceeded timelines of

project delivery. Therefore, teams having a mix of highly experienced and less experienced professional can fare better in software development teams.

Results from this study suggest that large functional heterogeneity in teams can be detrimental to the software projects as it can cause negative consequences for the productivity and the elapsed time of the software project. The moderation effects show that teams having high functional heterogeneity can have a reduced positive effect of the CMM – productivity relationship and reduced negative effect of the CMM – elapsed time relationship. Functional heterogeneity in software projects is common and is required in software teams, as development of software components require support from team that are experts in different functionalities such as database administrators, network administrators, computer operators, sometimes the end users and clients etc. Involving people from different functionalities during the development phase can lead to misunderstandings and communication issues as each person can understand the project intricacies at a different level which may cause differences in the opinion about the way the project is being developed. Previous studies also support that functional heterogeneity drives conflict as people have different understanding of the task priorities, assumptions about future events and understanding of various options about preforming tasks thus leading to task conflict and ultimately affecting the performance (Gebert, Boerner, & Kearney, 2006; Pelled, Eisenhardt, & Xin, 1999). Although functional heterogeneity has benefits in the form of information sharing and creativity empowerment (Srikanth, Harvey, & Peterson, 2016), it can reduce the productivity and increase the elapsed time of the project. Therefore, involving people from different functional diversities may not be a good idea as it can hamper the success rate of software projects.

To sum up, the study highlights the importance of human factors in software projects developed in CMM level software development organizations. The study suggests that, although software process initiatives help streamline the software development process, people factors can influence the software development project outcomes.

1.10 Managerial Implications

This research has important implications for managers with regard to staffing in the CMM level software development organizations. The results of the study are a step towards developing a theory for IS team staffing and human resource management in the IS context. Findings from the study suggest that people factor along with the process factors also play a significant role in the success of software development projects. While process controls applied through the software process improvement initiatives such as CMM/CMMI can enhance the productivity and reduce the overall time for development, improper staffing can be detrimental to the software project success. The study suggests software development managers to take utmost care about staffing of the employees. They have to consider human and team factors as people play a critical role in the software development process.

Software managers have the need to consider a number of factors in the selection criteria of their development team members among which work experience and functional expertise can greatly determine the software project success as identified from the study. Software managers also have the need to keep the turnover rate in the teams to a minimum since it can influence the project outcomes negatively. Software development managers need to plan and assess the risks and benefits of having experienced staff members, having more turnover rate, and the existence of functional heterogeneity in a development team. This study suggests the managers to include equal proportions of highly experienced and less experienced professionals in their teams to keep the team work balanced. The skillset of the highly experienced members is very essential in resolving the bottlenecks during the development process, however, a team consisting of only highly experienced professionals may not result in success as it can exceed the project elapsed time of the project due to their possible involvement in many projects as their expertise may be required in many projects. Whereas less experienced professional can keep themselves focused as they have the urge to learn new things and usually have a thrust to showcase their abilities. This urge can benefit the software projects by completing the projects on time.

The study also suggests the managers to keep the team functional heterogeneity levels at minimum levels. Although heterogeneity can enhance creativity, it can be unfavorable to the project outcomes. The study encourages the managers to monitor and to keep track of the project activities and suggest them to eliminate non-performing or underperforming employees to keep the project outcomes positive since employee turnover in CMM level projects is not of big concern. In turn, it can benefit the software projects by bringing in fresh talent and better performing employees. Finally, the managers should take people factors into consideration to fulfill the objectives and goals of the software organizations.

1.11 Conclusion

The study makes significant contributions to software process maturity research and practice. This study explores maturity of software development along the team management perspective by empirically testing the relationship between process maturity, project outcomes, and team characteristics. It reinforces the fact that team is an important building block of the software development projects. The team factors help in building a cohesive software development project team that can leverage on the organizational process to attain the projects' mission of delivering projects with high productivity in less time, within the budgets and with high quality. CMM level projects benefit from teams having both high and less experienced professionals, less functional heterogeneity, and more turnover. These factors help to achieve high productivity, as well as to deliver a project in less time in where higher CMM level projects can benefit from less number of team levels and less experienced team members' .Therefore, managers of the software development teams must support the teams and foster a suitable environment. It can no longer be said that a high maturity process in place is sufficient for a highly successful project but people management with team management skills, and senior management support is also equally important.

1.12 References

Abdel-Hamid, T. K., & Madnick, S. E. (1989). Lessons learned from modeling the dynamics of software development. *Communications of the ACM, 32*(12), 1426-1438.

Adolph, S., Kruchten, P., & Hall, W. (2012). Reconciling perspectives: A grounded theory of how people manage the process of software development. *Journal of Systems and Software, 85*(6), 1269-1286.

Agrawal, M., & Chari, K. (2007). Software effort, quality, and cycle time: A study of CMM level 5 projects. *Software Engineering, IEEE Transactions On, 33*(3), 145-156.

Agrawal, N. M., Khatri, N., & Srinivasan, R. (2012). Managing growth: Human resource management challenges facing the indian software industry. *Journal of World Business, 47*(2), 159-166.

Aiken, L. S., West, S. G., & Reno, R. R. (1991). *Multiple regression: Testing and interpreting interactions* Sage.

Al-Ahmad, W., Al-Fagih, K., Khanfar, K., Alsamara, K., Abuleil, S., & Abu-Salem, H. (2009). A taxonomy of an IT project failure: Root causes. *International Management Review, 5*(1), 93.

Aleti, A., Buhnova, B., Grunske, L., Koziolek, A., & Meedeniya, I. (2013). Software architecture optimization methods: A systematic literature review. *Software Engineering, IEEE Transactions On, 39*(5), 658-683.

Alter, S. (2013). Work system theory: Overview of core concepts, extensions, and challenges for the future. *Journal of the Association for Information Systems, 14*(2), 72.

Alves, R., Valente, P., & Nunes, N. J. (2013). Improving software effort estimation with human-centric models: A comparison of UCP and iUCP accuracy. Paper presented at the *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems,* 287-296.

André, M., Baldoquín, M. G., & Acuña, S. T. (2011). Formal model for assigning human resources to teams in software projects. *Information and Software Technology, 53*(3), 259-275.

Athavale, S., & Balaraman, V. (2013). Human behavioral modeling for enhanced software project management. Paper presented at the *7th International Conference on Software Engineering,* 15-17.

Athreye, S. S. (2004). "Role of transnational corporations in the evolution of a high-tech industry: The case of india's software Industry"—A comment. *World Development, 32*(3), 555-560.

Awad, M. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia,*

Baddoo, N., & Hall, T. (2002). Motivators of software process improvement: An analysis of practitioners' views. *Journal of Systems and Software, 62*(2), 85-96.

Baggen, R., Correia, J. P., Schill, K., & Visser, J. (2012). Standardized code quality benchmarking for improving software maintainability. *Software Quality Journal, 20*(2), 287-307.

Barki, H., Rivard, S., & Talbot, J. (1993). Toward an assessment of software development risk. *Journal of Management Information Systems, 10*(2), 203-225.

Baron, R. M., & Kenny, D. A. (1986). The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of Personality and Social Psychology, 51*(6), 1173.

Basha, S., & Ponnurangam, D. (2010). Analysis of empirical software effort estimation models. *arXiv Preprint arXiv:1004.1239,*

Basili, V. R., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., . . . Trendowicz, A. (2013). Linking software development and business strategy through measurement. *arXiv Preprint arXiv:1311.6224,*

Baxter, G., & Sommerville, I. (2011). Socio-technical systems: From design methods to systems engineering. *Interacting with Computers, 23*(1), 4-17.

Bayraktaroglu, A., Calisir, F., & Gumussoy, C. (2009). Usability and functionality: A comparison of project managers' and potential users' evaluations. Paper presented at the *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference On,* 2019-2023.

Beaver, J. M., & Schiavone, G. A. (2006). The effects of development team skill on software product quality. *ACM SIGSOFT Software Engineering Notes, 31*(3), 1-5.

Becerril, M. G. G. (2011). Reengineering a cloud computing platform using model based technologies.

Bhadury, J., Mighty, E. J., & Damar, H. (2000). Maximizing workforce diversity in project teams: A network flow approach. *Omega, 28*(2), 143-153.

Bloch, M., Blumberg, S., & Laartz, J. (2012). Delivering large-scale IT projects on time, on budget, and on value. *Harvard Business Review,*

Boehm, B. W. (1987). Improving software productivity. Paper presented at the *Computer,*

Bostrom, R. P., & Heinen, J. S. (1977). MIS problems and failures: A socio-technical perspective, part II: The application of socio-technical theory. *MIS Quarterly,* , 11-28.

Brooks, F. P. (1975). *The mythical man-month* Addison-Wesley Reading, MA.

Cappelli, P. (2008). Talent management for the twenty-first century. *Harvard Business Review, 86*(3), 74.

Capretz, L. F., & Ahmed, F. (2010). Making sense of software development and personality types. *IT Professional, 12*(1), 6-13.

Casey, V., Deshpande, S., & Richardson, I. (2010). Outsourcing software development the remote project manager's perspective. Paper presented at the

Cataldo, M., Herbsleb, J. D., & Carley, K. M. (2008). Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity. Paper presented at the *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement,* 2-11.

Chappell, D. (2012). Business value of software quality. Paper presented at the *North America Quest Conference& Expo 2012–Quality Engineered Software &Testing,*

Charette, R. N. (2005). Why software fails. *IEEE Spectrum, 42*(9), 36.

Chatzipetrou, P., Papatheocharous, E., Angelis, L., & Andreou, A. S. (2015). A multivariate statistical framework for the analysis of software effort phase distribution. *Information and Software Technology, 59*, 149-169.

Clarke, P., & O'Connor, R. V. (2012a). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology, 54*(5), 433-447.

Cohen, S. G., & Bailey, D. E. (1997). What makes teams work: Group effectiveness research from the shop floor to the executive suite. *Journal of Management, 23*(3), 239-290.

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., Misra, S., & García-Peñalvo, F. J. (2012). Analyzing human resource management practices within the GSD context. *Journal of Global Information Technology Management, 15*(3), 30-54.

Conchúir, E. Ó, Ågerfalk, P. J., Olsson, H. H., & Fitzgerald, B. (2009). Global software development: Where are the benefits? *Communications of the ACM, 52*(8), 127-131.

Coronel, C., & Morris, S. (2016). *Database systems: Design, implementation, & management* Cengage Learning.

Corral, L., Sillitti, A., & Succi, G. (2013). Software development processes for mobile systems: Is agile really taking over the business? Paper presented at the *Engineering of Mobile-Enabled Systems (MOBS), 2013 1st International Workshop on The,* 19-24.

Cronin, M. A., & Weingart, L. R. (2007). Representational gaps, information processing, and conflict in functionally diverse teams. *Academy of Management Review, 32*(3), 761-773.

Cuenca, L., Boza, A., Alemany, M., & Trienekens, J. J. (2013). Structural elements of coordination mechanisms in collaborative planning processes and their assessment through maturity models: Application to a ceramic tile company. *Computers in Industry, 64*(8), 898-911.

Cuevas, G., Calvo-Manzano, J. A., & García, I. (2014). Some key topics to be considered in software process improvement. *Agile Estimation Techniques and Innovative Approaches to Software Process Improvement,* , 119.

Cugola, G., & Ghezzi, C. (1998). Software processes: A retrospective and a path to the future. *Software Process: Improvement and Practice, 4*(3), 101-123.

Curtis, B., Hefley, W. E., & Miller, S. A. (2009). *People CMM: A framework for human capital management* Pearson Education.

Deephouse, C., Mukhopadhyay, T., Goldenson, D. R., & Kellner, M. I. (1995). Software processes and project performance. *Journal of Management Information Systems, 12*(3), 187-205.

Diaz, M., & King, J. (2002). How CMM impacts quality, productivity, rework, and the bottom line. *CrossTalk, 15*(3), 9-14.

Diaz, M., & Sligo, J. (1997). How software process improvement helped motorola. *IEEE Software, 14*(5), 75.

Dion, R. (1993). Process improvement and the corporate balance sheet. *IEEE Software, 10*(4), 28-35.

Dwivedi, Y. K., Ravichandran, K., Williams, M. D., Miller, S., Lal, B., Antony, G. V., & Kartik, M. (2013). IS/IT project failures: A review of the extant literature for deriving a taxonomy of failure factors. *Grand successes and failures in IT. public and private sectors* (pp. 73-88) Springer.

Easton, G. S., & Rosenzweig, E. D. (2012). The role of experience in six sigma project success: An empirical analysis of improvement projects. *Journal of Operations Management, 30*(7), 481-493.

Ebert, C., Murthy, B. K., & Jha, N. N. (2008). Managing risks in global software engineering: Principles and practices. Paper presented at the *2008 IEEE International Conference on Global Software Engineering,* 131-140.

El Emam, K., & Koru, A. G. (2008). A replicated survey of IT software project failures. *Software, IEEE, 25*(5), 84-90.

Elkhatib, Y., Blair, G. S., & Surajbali, B. (2013). Experiences of using a hybrid cloud to construct an environmental virtual observatory. Paper presented at the *Proceedings of the 3rd International Workshop on Cloud Data and Platforms,* 13-18.

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management, 16*(4), 88.

Espinosa, J. A., Slaughter, S. A., Kraut, R. E., & Herbsleb, J. D. (2007). Familiarity, complexity, and team performance in geographically distributed software development. *Organization Science, 18*(4), 613-630.

F. Hair Jr, J., Sarstedt, M., Hopkins, L., & G. Kuppelwieser, V. (2014). Partial least squares structural equation modeling (PLS-SEM) an emerging tool in business research. *European Business Review, 26*(2), 106-121.

Faraj, S., & Sproull, L. (2000). Coordinating expertise in software development teams. *Management Science, 46*(12), 1554-1568.

Feldt, R., Cheng, C., Gorschek, T., Moinul Islam, A., Unterkalmsteiner, M., & Permadi, R. (2012). Evaluation and measurement of software process Improvement—A systematic literature review.

Fernández-Diego, M., & González-Ladrón-de-Guevara, F. (2014). Potential and limitations of the ISBSG dataset in enhancing software engineering research: A mapping review. *Information and Software Technology, 56*(6), 527-544.

Fong Boh, W., Slaughter, S. A., & Espinosa, J. A. (2007). Learning from experience in software development: A multilevel analysis. *Management Science, 53*(8), 1315-1331.

Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research,* , 39-50.

FP Jr, B. (1987). No silver bullet essence and accidents of software engineering. *Computer,* (4), 10-19.

Fuggetta, A., & Di Nitto, E. (2014). Software process. Paper presented at the *Proceedings of the on Future of Software Engineering,* 1-12.

Galin, D., & Avrahami, M. (2006). Are CMM program investments beneficial? analyzing past studies. *IEEE Software, 23*(6), 81.

Garcia-Diaz, N., Lopez-Martin, C., & Chavoya, A. (2013). A comparative study of two fuzzy logic models for software development effort estimation. *Procedia Technology, 7*, 305-314.

Gaudin, S. (2003). Study: Many major IT projects still fail. *Datamation, 16*(1), 1-2.

Gebauer, J., Kline, D. M., & He, L. (2011). Password security risk versus effort: An exploratory study on user-perceived risk and the intention to use online applications. *Journal of Information Systems Applied Research, 4*(2), 52.

Gebert, D., Boerner, S., & Kearney, E. (2006). Cross-functionality and innovation in new product development teams: A dilemmatic structure and its consequences for the management of diversity. *European Journal of Work and Organizational Psychology, 15*(4), 431-458.

Gefen, D., Straub, D., & Boudreau, M. (2000). Structural equation modeling and regression: Guidelines for research practice. *Communications of the Association for Information Systems, 4*(1), 7.

Gillies, A. (2011). *Software quality: Theory and management* Lulu. com.

Goguen, J. (1999). Tossing algebraic flowers down the great divide. *People and Ideas in Theoretical Computer Science,* , 93-129.

Goles, T., Hawk, S., & Kaiser, K. M. (2008). Information technology workforce skills: The software and IT services provider perspective. *Information Systems Frontiers, 10*(2), 179-194.

Gopal, A., & Gosain, S. (2010). Research note-the role of organizational controls and boundary spanning in software development outsourcing: Implications for project performance. *Information Systems Research, 21*(4), 960-982.

Gopal, A., & Koka, B. R. (2012). The asymmetric benefits of relational flexibility: Evidence from software development outsourcing. *Mis Quarterly, 36*(2), 553-576.

Gopal, A., Sivaramakrishnan, K., Krishnan, M. S., & Mukhopadhyay, T. (2003). Contracts in offshore software development: An empirical analysis. *Management Science, 49*(12), 1671-1683.

Gordon, P. (1999). To err is human, to estimate, divine. *Information Week, 711*, 65-72.

Gorla, N., & Lam, Y. W. (2004). Who should work with whom?: Building effective software project teams. *Communications of the ACM, 47*(6), 79-82.

Gorla, N., & Lin, S. (2010). Determinants of software quality: A survey of information systems project managers. *Information and Software Technology, 52*(6), 602-610.

Guinan, P. J., Cooprider, J. G., & Faraj, S. (1998). Enabling software development team performance during requirements definition: A behavioral versus technical approach. *Information Systems Research, 9*(2), 101-125.

Guzzo, R. A., & Dickson, M. W. (1996). Teams in organizations: Recent research on performance and effectiveness. *Annual Review of Psychology, 47*(1), 307-338.

Hair, J. F., Ringle, C. M., & Sarstedt, M. (2011). PLS-SEM: Indeed a silver bullet. *Journal of Marketing Theory and Practice, 19*(2), 139-152.

Hair, J. F., Anderson, R. E., Tatham, R. L., & William, C. (1998). Black (1998), multivariate data analysis.

Haley, T. J. (1996). Software process improvement at raytheon. *IEEE Software, 13*(6), 33.

Hambrick, D. C., Cho, T. S., & Chen, M. (1996). The influence of top management team heterogeneity on firms' competitive moves. *Administrative Science Quarterly,* , 659-684.

Han, W. (2014). Validating differential relationships between risk categories and project performance as perceived by managers. *Empirical Software Engineering, 19*(6), 1956-1966.

Hanssen, G. K. (2012). A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. *Journal of Systems and Software, 85*(7), 1455-1466.

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science, 46*(4), 451-466.

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

Hoegl, M., & Parboteeah, K. P. (2007). Creativity in innovative projects: How teamwork matters. *Journal of Engineering and Technology Management, 24*(1), 148-166.

Houston, D. X., Mackulak, G. T., & Collofello, J. S. (2001). Stochastic simulation of risk factor potential effects for software development risk management. *Journal of Systems and Software, 59*(3), 247-257.

Hu, Y., Zhang, X., Ngai, E., Cai, R., & Liu, M. (2013). Software project risk analysis using bayesian networks with causality constraints. *Decision Support Systems, 56*, 439-449.

Huang, S., & Chiu, N. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and Software Technology, 48*(11), 1034-1045.

Jiang, Z., & Naudé, P. (2007). An examination of the factors influencing software development effort. *International Journal of Computer Information and Systems Science and Engineering, 1*(3), 182-191.

Jodpimai, P., Sophatsathit, P., & Lursinsap, C. (2010). Estimating software effort with minimum features using neural functional approximation. Paper presented at the *2010 International Conference on Computational Science and its Applications,* 266-273.

Jones, C. (2006). Social and technical reasons for software project failures. *STSC Cr0ssTalk June,*

Jørgensen, M. (2010). Identification of more risks can lead to increased over-optimism of and over-confidence in software development effort estimates. *Information and Software Technology, 52*(5), 506-516.

Jørgensen, M. (2014). Failure factors of small software projects at a global outsourcing marketplace. *Journal of Systems and Software, 92*, 157-169.

Jorgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *Software Engineering, IEEE Transactions On, 33*(1), 33-53.

Jørgensen, M., & Sjøberg, D. I. (2004). The impact of customer expectation on software development effort estimates. *International Journal of Project Management, 22*(4), 317-325.

Joshi, A., & Roh, H. (2009). The role of context in work team diversity research: A meta-analytic review. *Academy of Management Journal, 52*(3), 599-627.

Juristo, N., Moreno, A. M., & Sanchez-Segura, M. (2007). Analysing the impact of usability on software design. *Journal of Systems and Software, 80*(9), 1506-1516.

Kang, H., Yang, H., & Rowley, C. (2006). Factors in team effectiveness: Cognitive and demographic similarities of software development team members. *Human Relations, 59*(12), 1681-1710.

Kaur, R., & Sengupta, J. (2013). Software process models and analysis on failure of software development projects. *arXiv Preprint arXiv:1306.1068,*

Khan, A. I., Qurashi, R. J., & Khan, U. A. (2011). A comprehensive study of commonly practiced heavy and light weight software methodologies. *arXiv Preprint arXiv:1111.3001,*

Khatibi, V., Jawawi, D. N., & Khatibi, E. (2012). Investigating the effect of using methodology on development effort in software projects.

Kitchenham, B., & Pfleeger, S. L. (1996). Software quality: The elusive target. *IEEE Software, 13*(1), 12.

Kock, N. (2015). WarpPLS 5.0 user manual. *Laredo, TX: ScriptWarp Systems.*

Kong, E., Chadee, D., & Raman, R. (2013). Managing indian IT professionals for global competitiveness: The role of human resource practices in developing knowledge and learning capabilities for innovation. *Knowledge Management Research & Practice, 11*(4), 334-345.

Korrapati, R., & Eedara, V. S. (2010). A study of the relationship between software project success and employee job satisfaction. Paper presented at the *Allied Academies International Conference. Academy of Information and Management Sciences. Proceedings, , 14*(1) 22.

Krishnan, M. S., Kriebel, C. H., Kekre, S., & Mukhopadhyay, T. (2000). An empirical analysis of productivity and quality in software products. *Management Science, 46*(6), 745-759.

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software,* (6), 18-21.

Krueger, C. W. (2007). The 3-tiered methodology: Pragmatic insights from new generation software product lines. Paper presented at the *Software Product Line Conference, 2007. SPLC 2007. 11th International,* 97-106.

Kulpa, M. (2007). Why should I use the people CMM. *CROSSTALK.the Journal of Defense Software Engineering, 20*(11), 19-22.

Kumaresh, S., & Baskaran, R. (2010). Defect analysis and prevention for software process quality improvement. *International Journal of Computer Applications, 8*(7)

Lacity, M. C., & Hirschheim, R. (1993). The information systems outsourcing bandwagon. *Sloan Management Review, 35*(1), 73.

Langer, N., Slaughter, S. A., & Mukhopadhyay, T. (2014). Project managers' practical intelligence and project performance in software offshore outsourcing: A field study. *Information Systems Research, 25*(2), 364-384.

Laporte, C. Y., & Trudel, S. (1998). Addressing the people issues of process improvement activities at oerlikon aerospace. *Software Process: Improvement and Practice, 4*(4), 187-198.

Lee, G., DeLone, W., & Espinosa, J. A. (2006). Ambidextrous coping strategies in globally distributed software development projects. *Communications of the ACM, 49*(10), 35-40.

Lee, G., Espinosa, J. A., & DeLone, W. H. (2013). Task environment complexity, global team dispersion, process capabilities, and coordination in software development. *IEEE Transactions on Software Engineering, 39*(12), 1753-1771.

Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived causes of software project failures–An analysis of their relationships. *Information and Software Technology, 56*(6), 623-643.

Lending, D., & Chervany, N. L. (1998). The use of CASE tools. Paper presented at the *Proceedings of the 1998 ACM SIGCPR Conference on Computer Personnel Research,* 49-58.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., . . . Zelkowitz, M. (2002). Empirical findings in agile methods. Paper presented at the *Conference on Extreme Programming and Agile Methods,* 197-207.

Lowry, P. B., & Gaskin, J. (2014). Partial least squares (PLS) structural equation modeling (SEM) for building and testing behavioral causal theory: When to choose it and how to use it. *Professional Communication, IEEE Transactions On, 57*(2), 123-146.

Lu, P., Song, X., & Song, Y. (2012). An empirical analysis of requirements uncertainty, task uncertainty and software project performance. *Journal of Emerging Trends in Economics and Management Sciences, 3*(5), 559.

Lu, Y., Xiang, C., Wang, B., & Wang, X. (2011). What affects information systems development team performance? an exploratory study from the perspective of combined socio-technical theory and coordination theory. *Computers in Human Behavior, 27*(2), 811-822.

Luftmann, J., & Kempaiah, R. (2008). Key issues for IT executives 2007. *MIS Quarterly Executive, 7*(2)

Luna-Reyes, L. F., Zhang, J., Gil-García, J. R., & Cresswell, A. M. (2005). Information systems development as emergent socio-technical change: A practice approach. *European Journal of Information Systems, 14*(1), 93-105.

Magdaleno, A. M., Werner, C. M. L., & De Araujo, R. M. (2012). Reconciling software development models: A quasi-systematic review. *Journal of Systems and Software, 85*(2), 351-369.

Magjuka, R. J., & Baldwin, T. T. (1991). Team-based employee involvement programs: Effects of design and administration. *Personnel Psychology, 44*(4), 793-812.

Maloney, M. M., & Zellmer-Bruhn, M. E. (2006). Building bridges, windows and cultures: Mediating mechanisms between team heterogeneity and performance in global teams. *Management International Review, 46*(6), 697-720.

Marasco, J., & Ravenflow, C. (2006). Software development productivity and project success rates: Are we attacking the right problem. *The Rational Edge, February, 15*

Martin, N. L., Pearson, J. M., & Furumo, K. A. (2005). IS project management: Size, complexity, practices and the project management office. Paper presented at the *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference On,* 234b-234b.

Michel, J. G., & Hambrick, D. C. (1992). Diversification posture and top management team characteristics. *Academy of Management Journal, 35*(1), 9-37.

Miller, F. P., Vandome, A. F., & McBrewster, J. (2010). Fourth-generation programming language: Programming language, history of computer science, third-generation programming language, fifth-generation programming... problem solving, systems engineering.

Milliken, F. J., & Martins, L. L. (1996). Searching for common threads: Understanding the multiple effects of diversity in organizational groups. *Academy of Management Review, 21*(2), 402-433.

Mullaly, M. (2006). Longitudinal analysis of project management maturity. *Project Management Journal, 37*(3), 62.

Mundra, A., Misra, S., & Dhawale, C. A. (2013). Practical scrum-scrum team: Way to produce successful and quality software. Paper presented at the *Computational Science and its Applications (ICCSA), 2013 13th International Conference On,* 119-123.

Murray, A. I. (1989). Top management group heterogeneity and firm performance. *Strategic Management Journal, 10*(S1), 125-141.

Nasir, M., & Sahibuddin, S. (2011). Addressing a critical success factor for software projects: A multi-round delphi study of TSP. *International Journal of the Physical Sciences, 6*(5), 1213-1232.

Nasir, M. H. N., & Sahibuddin, S. (2011). Critical success factors for software projects: A comparative study. *Scientific Research and Essays, 6*(10), 2174-2186.

Nelson, R. R., & Morris, M. G. (2014). IT project estimation: Contemporary practices and management guidelines. *MIS Quarterly Executive, 13*(1)

Ngwenyama, O., & Nielsen, P. A. (2003). Competing values in software process improvement: An assumption analysis of CMM from an organizational culture perspective. *IEEE Transactions on Engineering Management, 50*(1), 100-112.

Nidumolu, S. (1995). The effect of coordination and uncertainty on software project performance: Residual performance risk as an intervening variable. *Information Systems Research, 6*(3), 191-219.

Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company: How japanese companies create the dynamics of innovation* Oxford university press.

Norman, D. A. (1993). *Things that make us smart: Defending human attributes in the age of the machine* Basic Books.

Nunnally, J. C., & Bernstein, I. (1994). The assessment of reliability. *Psychometric Theory, 3*(1), 248-292.

Palvia, S. C., Sharma, R. S., & Conrath, D. W. (2001). A socio-technical framework for quality assessment of computer information systems. *Industrial Management & Data Systems, 101*(5), 237-251.

Paulk, M. C. (2009). A history of the capability maturity model for software. *ASQ Software Quality Professional, 12*(1), 5-19.

Pelled, L. H., Eisenhardt, K. M., & Xin, K. R. (1999). Exploring the black box: An analysis of work group diversity, conflict and performance. *Administrative Science Quarterly, 44*(1), 1-28.

Pendharkar, P. C., & Rodger, J. A. (2007). An empirical study of the impact of team size on software development effort. *Information Technology and Management, 8*(4), 253-262.

Pendharkar, P. C., Subramanian, G. H., & Rodger, J. A. (2005). A probabilistic model for predicting software development effort. *Software Engineering, IEEE Transactions On, 31*(7), 615-624.

Ply, J. K., Moore, J. E., Williams, C. K., & Thatcher, J. B. (2012). IS employee attitudes and perceptions at varying levels of software process maturity. *MIS Quarterly, 36*(2), 601-624.

Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering, 4*(4), 345.

Radliński, Ł. (2012). Towards expert-based modelling of integrated software quality. *Journal of Theoretical and Applied Computer Science, 6*(2), 13-26.

Ramanujan, S., & Kesh, S. (2004). Comparison of knowledge management and CMM/CMMI implementation. *Journal of American Academy of Business, 4*(1/2), 271-275.

Ramasubbu, N., Cataldo, M., Balan, R. K., & Herbsleb, J. D. (2011). Configuring global software teams: A multi-company analysis of project productivity, quality, and profits. Paper presented at the *Proceedings of the 33rd International Conference on Software Engineering,* 261-270.

Ramasubbu, N., Mithas, S., Krishnan, M. S., & Kemerer, C. F. (2008). Work dispersion, process-based learning, and offshore software development performance. *MIS Quarterly,* , 437-458.

Ravichandran, T., & Rai, A. (2000). Quality management in systems development: An organizational system perspective. *MIS Quarterly,* , 381-415.

Ray, B., Posnett, D., Filkov, V., & Devanbu, P. (2014). A large scale study of programming languages and code quality in github. Paper presented at the *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering,* 155-165.

Reel, J. S. (1999). Critical success factors in software projects. *IEEE Software, 16*(3), 18-23.

Rodriguez, D., Sicilia, M., Garcia, E., & Harrison, R. (2012). Empirical findings on team size and productivity in software development. *Journal of Systems and Software, 85*(3), 562-570.

Rosson, M. B. (1996). Human factors in programming and software development. *ACM Computing Surveys (CSUR), 28*(1), 193-195.

Sadikov, V., & Pidkameny, W. (2014). Complete separation of the 3 tiers-divide and conquer. *arXiv Preprint arXiv:1405.1618,*

Samadhiya, D., Wang, S., & Chen, D. (2010). Quality models: Role and value in software engineering. Paper presented at the *Software Technology and Engineering (ICSTE), 2010 2nd International Conference On, , 1* V1-320-V1-324.

Sandhu, G. S., & Salaria, D. S. (2014). A bayesian network model of the particle swarm optimization for software effort estimation. *International Journal of Computer Applications, 96*(4)

Sarma, A., & Herbsleb, J. (2008). Using development experience to calculate congruence. Paper presented at the *Workshop on Socio-Technical Congruence, in Conjunction with the International Conference on Software Engineering,*

Savalei, V., & Bentler, P. M. (2010). Structural equation modeling. *Corsini Encyclopedia of Psychology,*

Scacchi, W., & Hurley, D. (1995). Understanding software productivity. *Software Engineering and Knowledge Engineering: Trends for the Next Decade, 4*, 273-316.

Schach, S. R. (2002). *Object-oriented and classical software engineering* McGraw-Hill New York.

Schmidt, D. C. (2006). Model-driven engineering. *Computer-Ieee Computer Society-, 39*(2), 25.

Siau, K., Tan, X., & Sheng, H. (2010). Important characteristics of software development team members: An empirical investigation using repertory grid. *Information Systems Journal, 20*(6), 563-580.

Silva, F. S., Soares, F. S. F., Peres, A. L., de Azevedo, I. M., Vasconcelos, A. P. L., Kamei, F. K., & de Lemos Meira, Silvio Romero. (2015). Using CMMI together with agile software development: A systematic review. *Information and Software Technology, 58*, 20-43.

Šmite, D., Wohlin, C., Gorschek, T., & Feldt, R. (2010). Empirical evidence in global software engineering: A systematic review. *Empirical Software Engineering, 15*(1), 91-118.

Smith, R. K., Hale, J. E., & Parrish, A. S. (2001). An empirical study using task assignment patterns to improve the accuracy of software effort estimation. *IEEE Transactions on Software Engineering, 27*(3), 264-271.

Söderland, J., Geraldi, J., Müller, R., & Jugdev, K. (2012). Critical success factors in projects: Pinto, slevin, and prescott-the elucidation of project success. *International Journal of Managing Projects in Business, 5*(4), 757-775.

Somech, A., & Drach-Zahavy, A. (2013). Translating team creativity to innovation implementation the role of team composition and climate for innovation. *Journal of Management, 39*(3), 684-708.

Srikanth, K., Harvey, S., & Peterson, R. (2016). A dynamic perspective on diverse teams: Moving from the dual-process model to a dynamic coordination-based model of diverse team performance. *The Academy of Management Annals, 10*(1), 453-493.

Staples, M., & Niazi, M. (2008). Systematic review of organizational motivations for adopting CMM-based SPI. *Information and Software Technology, 50*(7), 605-620.

Staron, M. (2012). Critical role of measures in decision processes: Managerial and technical measures in the context of large software development organizations. *Information and Software Technology, 54*(8), 887-899.

Subramanian, G. H., Jiang, J. J., & Klein, G. (2007). Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. *Journal of Systems and Software, 80*(4), 616-627.

Sudeikat, J., Braubach, L., Pokahr, A., & Lamersdorf, W. (2004). Evaluation of agent–oriented software methodologies–examination of the gap between modeling and platform. *Agent-oriented software engineering V* (pp. 126-141) Springer.

Syverson, C. (2011). What determines productivity? *Journal of Economic Literature, 49*(2), 326-365.

Talwar, V., Wu, Q., Pu, C., Yan, W., Jung, G., & Milojicic, D. (2005). Comparison of approaches to service deployment. Paper presented at the *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference On,* 543-552.

Team, C. P. (2002). Capability maturity model® integration (CMMI SM), version 1.1. *CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1),*

Thamhain, H. J. (2004). Leading technology-based project teams. *Engineering Management Journal, 16*(2), 35-43.

Thomas-Hunt, M. C., Ogden, T. Y., & Neale, M. A. (2003). Who's really sharing? effects of social and expert status on knowledge exchange within groups. *Management Science, 49*(4), 464-477.

Tierno, I. A., & Nunes, D. J. (2012). Assessment of automatically built bayesian networks in software effort prediction. Paper presented at the *CIbSE,* 196-209.

Trist, E., & Bamforth, K. (1951). Some social and psychological consequences of the longwall method. *Human Relations, 4*(3), 3-38.

Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., & Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS), 3*(1), 1.

Verner, J. M., Babar, M. A., Cerpa, N., Hall, T., & Beecham, S. (2014). Factors that motivate software engineering teams: A four country empirical study. *Journal of Systems and Software, 92*, 115-127.

Wallace, L., Keil, M., & Rai, A. (2004). How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model. *Decision Sciences, 35*(2), 289-321.

Wi, H., Oh, S., Mun, J., & Jung, M. (2009). A team formation model based on knowledge and collaboration. *Expert Systems with Applications, 36*(5), 9121-9134.

Xia, W., & Lee, G. (2004). Grasping the complexity of IS development projects. *Communications of the ACM, 47*(5), 68-74.

Yoon, S. W., & Lim, D. H. (2010). Systemizing virtual learning and technologies by managing organizational competency and talents. *Advances in Developing Human Resources, 12*(6), 715-727.

Zimmermann, T., Premraj, R., & Zeller, A. (2007). Predicting defects for eclipse. Paper presented at the *Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop On,* 9-9.

Chapter 2: The Technological Determinants of Quality and Effort of Software Development Projects

2.1 Introduction

Many software projects are late, over budget, unpredictable and sometimes fail even before delivery (Dwivedi et al., 2013; Jørgensen, 2014; Verner, Babar, Cerpa, Hall, & Beecham, 2014). According to the Standish group chaos manifesto, in 2012, 39 % of software development projects are successful (delivered on time with full functionality), 43 % are challenged (late, over budget or delivered with less than the required features and functions) and 18 % are failed (cancelled or delivered but never used due to software failures) (Nelson & Morris, 2014).[1] The most common reason for schedule slippages, cost overruns or outright cancellation of software projects is that they contain too many bugs to operate successfully (Jones, 2006).

Despite the high percentage of challenged and failed software projects, software development activities never reduced given the increased need for information technology assets by end users (organizations, businesses, consumers, individuals) (Basili et al., 2013). However, delivering software products with less quality can reduce the business value of organizations for which the software has been built (Chappell, 2012). The damage caused by faulty software builds over time and can cause financial losses and also lower brand equity. In some cases, the consequences can be fatal to the end-users. One of the recent software disasters occurred in mid-January, 2016 for the Nest 'smart' thermostat (owned by Google) that allows users to monitor and adjust their thermostats on the smartphones. The application was hit with a software glitch and left users in the cold when a software update went wrong, forcing the devices batteries to drain and leaving it unavailable to control temperature. Software with defects not only affects the users that use it but also impacts the reputation of the IT group that develops it. Since software is important for every organization and end user, it is necessary for the software development organizations to find ways to reduce the failures and deliver quality software applications that are free from errors, and which, the modern businesses need. A survey of CIO's identifies improving IT quality as one

[1] https://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf

of the top concerns faced by the IT executives (Luftmann & Kempaiah, 2008). Hence, the study focusses on understanding the drivers of software development quality.

Similar to the quality aspect of software development, estimation of development effort is a crucial step in software development success as it increases the credibility of the client to the business enterprise (Basha & Ponnurangam, 2010). Inaccurate estimation of effort can be detrimental to the software quality and eventually the organization's reputation is at stake (Huang & Chiu, 2006). Software effort estimation depends on the various drivers of the software effort derived from objective information obtained from the historical project data and subjective evaluations from experts through judgment and analogies (Tierno & Nunes, 2012). Various effort estimation models exist for this purpose. To have a proper understanding of the effort estimation models and to prepare accurate estimates, it is important to understand the impact of the drivers of software development effort.

One of the leading factors that can lead to software project failures are the technical factors upon which the software is developed (Jones, 2006). Software technologies used in the project environment can have a significant impact on the software project outcomes since development and deployment of software systems is a multidimensional process where people and technology are interconnected (Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius, 2014). El Emam & Koru (2008) performed a survey to identify the reasons for software project failures and found that failures due to technology is one of the critical reason, as some of the technologies are too new, and they do not work as expected. Moreover, technical factors play a critical role in the decision making process of software development process such as release plans, selection of test strategies etc. (Staron, 2012). Hence, in this study, the important technological determinants of software development quality and effort are identified, a research model is proposed and hypotheses are generated about the technical factors. The model is empirically tested using structural equation modeling (SEM) analysis to test the impacts of technological determinants of software development quality and development effort with 1567 projects developed between 1994 and 2014 from ISBSG database.

This study, helps answer questions related to the type of technologies that can be used during the software development process, such as 1) should the manager invest in the latest programming language or use a particular kind of architecture for the development process? 2) Does using a particular type of architecture hold good for newer projects also? 3) In addition to analyzing the technical determinants, situational drivers are also included (Clarke & O'Connor, 2012) such as team size, project size in the research model to test their impact on software project quality and software development effort.

Statistical analysis about the causes of defects can help in defect prevention since the software development organizations can learn to avoid the kinds of errors and reduce the probability of software disasters (Jones, 2006). Although there have been studies performed about the determinants of quality and software effort of software projects, these studies have been specific to one particular country or one particular project. Few other research were focused on identifying the determinants of quality among the technical, social organizational factors based on but not a detailed analysis on each of these factors (Agrawal & Chari, 2007; Gorla & Lin, 2010). Empirical results on software quality and effort involving software development projects from multiple nations are rare. Quantitative analysis of the model and testing the proposed hypotheses will fill this gap by determining the relationship between the technical and situational factors that can impact the software development quality and effort. The unique features of the empirical analysis will be:

- Analysis of a large number of software development projects (> 1500 in number).
- Projects from multiple nations (> 20 in total).
- Projects covered over a large period (1991-2014).
- Focus on technical project features.

2.2 Theory and Literature Review

The software is embedded in many activities of day-to-day modern society and every year billions of dollars are lost in the US alone in software glitches. "Software Hall of Shame"

documents many IT projects that have failed (Charette, 2005). Software failures can cause human deaths in some situations (Software Bug, 2008). Software failures happen in all nations, all firms and have been with the software development process since its beginning. A prevention of failure strategy is necessary, but not many firms pay careful attention to it, probably due to limitations of resources and cost- and time constraints involved, among other things. It has been argued that the cost of fixing an error could be 100 times more if it is detected when implemented than when identified during the development stage. Therefore, it necessary to find ways to prevent software failures.

Developing software to meet the needs of the end users with quality, within budget and within time is an essential goal of the software organizations. Failure to meet any of these targets leads failure of the software projects (Al-Ahmad et al., 2009). Therefore, software quality and development effort are identified as important attributes that can measure the success/failure of software development projects.

Software Quality as defined by the US Department of Defense is "the degree to which the attributes of software enable it to perform its intended end use" (Gillies, 2011). Quality according to ISO 1986 standard suite is "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" (Gillies, 2011; Palvia, Sharma, & Conrath, 2001; Samadhiya, Wang, & Chen, 2010). Many factors contribute to the software error and thus quality. Some of these are (Charette, 2005; Krishnan, Kriebel, Kekre, & Mukhopadhyay, 2000): lack of better processes, lack of investments in the initial stages, badly defined system requirements, unrealistic project goals, lack of personal capability, poor communication among stakeholders, use of immature technology, poor development practices, etc. to name a few. Based on these two definitions that were stated, post implementation success and quality evaluations of software can be considered synonymous with each other (Palvia et al., 2001). Although software quality according to Chappell (2012) has three different dimensions: functional quality (the delivered product correctly performs the tasks as intended), structural quality (well-structured code) and process quality (quality of the development process). Functionality and usability are critical factors

of project success (Bayraktaroglu, Calisir, & Gumussoy, 2009; Juristo, Moreno, & Sanchez-Segura, 2007). Hence, post-implementation success/functional quality is deemed as important and has a greater weight in measuring the software success. Moreover, it is the most widely used metric for measuring the software development quality. Therefore, the study defines the quality of software as one which is measured by the number of defects reported in the first month of use of software—the lesser the number of defects in the software, the better the quality of the software is. Agrawal & Chari (2007) analyzed team and project related factors such as project size, the complexity of projects, manager experience, the skill level of the team, etc. that can impact the quality and found that only team size is significant.

Estimation of software development effort is one of the critical approaches to contain the cost of the software projects (Pendharkar, Subramanian, & Rodger, 2005). Software development effort typically includes human effort hours spent on various activities in development; this can be called as a synonym for software the cost as cost of person hours is the dominant cost in development (Agrawal & Chari, 2007). Practitioners use various methods to estimate software effort which includes objective (historical project data) and subjective measures (expert judgment and analogies). They use empirical methods that are expert analogy based (Jørgensen & Sjøberg, 2004), Bayesian network approach based (Sandhu & Salaria, 2014), fuzzy logic based (Garcia-Diaz, Lopez-Martin, & Chavoya, 2013), neural network based (Jodpimai, Sophatsathit, & Lursinsap, 2010) etc. Agrawal & Chari (2007) analyzed various team and project related drivers of effort and found that only team size impacts the software effort. Jørgensen (2010) identified that, a thorough work on risk identification can result in accurate estimation of effort. Pendharkar et al. (2005) identified that case tool experience, type of case tool used and methodology are important factors for software effort estimation and developed a Bayesian effort forecasting model with historical data and subjective expert judgment. However, very few have analyzed about the type of impacts of the determinants of the effort. Understanding the factors that impact effort will help in accurate estimation of software effort and cost.

Software development as a discipline has grown substantially richer through practice-influencing research and proven practices. Prior research about development effort was mainly focused on identifying the factors that can impact the effort estimation through surveys and expert judgment. The identified factors have been used in the effort estimation models for the introduction, evaluation and comparison of the software effort estimation methods (Jorgensen & Shepperd, 2007). However, a significant understanding needs to be achieved about the impact of each type of factor in detail. In particular, there is a need to examine the relationships between project outcomes and various factors identified from prior literature due to the increased use of new technologies, best practices of development, etc. Hence this study quantitatively evaluates the effect of technological factors on software project outcomes as shown in Figure 2.1.

Figure 2.1: Effect of Technological Factors on Project Performance

2.2.1 Research Gap

Al-Ahmad et al. (2009) explored about the root causes of Information Technology (IT) project failures and created a taxonomy with six main factors: technology factors, project management factors, complexity/size factors, process factors, organizational factors are prominent ones. Nasir & Sahibuddin (2011) identified that the critical success factors for software development are people, process, and technology. Fuggetta & Di Nitto (2014) state that technological breakthroughs and market disruptions have created new challenges for software engineering researchers and practitioners. All the three research articles notify that technical facts play a vital role in the software development failures. The incredibly fast developments in the technologies, calls for a need to evaluate the impact of technological factors on the project outcomes. Therefore, the current research will focus on addressing the following the following research questions

Research questions

Research Question 1 (RQ1): How do technical factors in the software projects drive the quality of software projects?

Research Question 2 (RQ2): How do technical factors in the software projects drive the effort of software projects?

Review of the previous information systems literature suggest the important technological factors that can impact the quality and effort of software projects. In like with this the following hypotheses are proposed. In addition to this, prior research also identify that situational factors also play a critical role on the software development project outcomes. However, prior researchers tested the impact of situational drivers such as team size and project size with older data set. Hence in this study, the impact of situational drivers along with the technological drivers will be tested using the latest project data set.

2.3 Hypotheses

2.3.1 Drivers of Quality

The Programming Language Used (generation)

The effect of programming languages on quality has been a question of much debate for a very long time (Ray, Posnett, Filkov, & Devanbu, 2014). Ray et al. (2014) note that programming language can, not only, impact the coding process but also the properties of resulting artifact. There are many programming languages available such as 1gl (machine languages), 2gl (assembler languages), 3gl (high-level languages, procedural), 4gl (non- procedural) and 5gl (automated). The 5gl represents the latest generation language. Later generation languages come with updated and enhanced features which can speed up the development process and also reduce the complexities that older generation languages come with. The later generation languages overcome the problems associated with older generation languages helping the developers to create quality software products. There is also other side of the view that later generation languages may not always result in success. For instance, Kumaresh & Baskaran (2010) performed a root-cause analysis to identify

the causes of defects among five software projects and found that introduction of new programming languages during the development phase is one main reason. The authors state that proper training and prior intimation about the adoption of new programming languages should be known to the developers, without which the quality of the software product can be compromised. Sometimes the programming languages are well known to the developers, and suddenly a new programming language is introduced, or the same is released with modifications and ramifications causing difficulties for the maintenance (Becerril, 2011). Chatzipetrou, Papatheocharous, Angelis, & Andreou (2015) performed an analysis with data from 1500 software projects from the ISBSG data to analyze the software effort distribution between six phases. The authors found that experienced professionals prefer third generation languages for performance reasons, and they concentrate more on the design phase. However, the use of 4GL is increasing, to take advantage of the powerful CASE tools. Nevertheless, software applications developed using 4GL have a high risk of poor and error prone behavior. Hence more time is spent on the testing phase. Schach (2002) state that among 43 organizations that implemented 4GL, only 28 organizations reported that the benefits outweighed costs, and there were plenty of bad experiences and on average, 4gls were inefficient. Later generation languages can introduce more complexity in coding because of their non-procedural characteristic and hence more errors in the software design. On the other hand, languages such as C++, Java, and C # are sophisticated and have libraries that can be reused which may reduce error (Marasco & Ravenflow, 2006). Although there are mixed views about using later and older generation languages the study proposes that using later generation languages has comparatively more benefits compared to older generation languages. Hence,

H1: Projects using later generation languages result in high quality software products.

The Development Method Used

Development methodologies are frameworks that define the different activities that are involved in the development process of the systems and also describe how the activities will be performed. They make the development process disciplined by making software development

more predictable and more efficient (Awad, 2005). The development methods have evolved significantly: from structured design to object-oriented to agile methods in recent times. Software development organizations use various methodologies such as waterfall, agile, scrum, extreme programming, etc. for the development process. While traditional methods follow a detailed and step by step process, the agile methodologies strip away the heaviness of traditional software development practices and promote quick response to changing environments and user requirements (Erickson, Lyytinen, & Siau, 2005). Other methodologies used in software development are software process improvement (SPI) techniques, which are mainly focused on the production process rather than the product. However, SPI's work under the assumption that better process results in better quality products (Baggen, Correia, Schill, & Visser, 2012). One of the biggest problems in implementing software development methodologies over the years has been the attempted mismatch of culture and methodology (Highsmith & Cockburn, 2001). Jiang & Naudé (2007) performed an empirical investigation with ISBSG data and found that development methodology is a significant factor for the productivity of software projects. However, in this study, the impact of using a development methodology on the quality of software development is tested. To sum up, the methodologies help in enhancing the developing process by performing the development activities in a planned manner. They can positively impact the quality of the software product.

H2: *Projects using development methodologies result higher quality software projects.*

The Computer Aided Software Engineering Tools (CASE) Used

Methodology and tools are the two areas where most money has been spent in the past 25 years (Marasco & Ravenflow, 2006). CASE tools are software application programs that are used to provide automated assistance for software development activities; they are intended to reduce the time and cost of software development and enhance the quality of software systems developed (Coronel & Morris, 2016; Lending & Chervany, 1998). They can be divided into four broad categories based on their use in the software development: central repository tools (which provide

common, consistent information to all the development team members), upper case tools (used in planning, analysis and design), lower case tools (used in implementation testing and maintenance) and integrated case tools (used from requirements gathering to testing and maintenance). Since good developers are scarce, these tools are aimed at maximizing the productivity. They provide a central data encyclopedia to which all developers must refer, and they ensure that good practices are carried out in a consistent manner throughout the development (Gillies, 2011). Introducing CASE tools in the design cycle may decrease the errors in the software design and improve the quality. Nevertheless, case tools can improve the quality only if they are applied universally in the organizations (Gillies, 2011). Despite the benefits, CASE tools are not adopted widely in practice since their one size fits all graphical representation are too generic and non-customizable (Schmidt, 2006). One of the problems is that the graphical language representation for writing programs mapped poorly onto the underlying platforms. Considering the benefits provide by the CASE tools the following hypotheses is proposed.

H3: Projects using CASE tools result in higher quality software projects.

The Architecture (stand alone or others)

Software architecture helps structure the complex software systems and provides a blueprint for the software developers. Architecture plays a significant role is the development of large systems together with other developmental activities (Kruchten, Nord, & Ozkaya, 2012). The decisions made during architecture design have significant implications for economic and quality goals of software products (Aleti, Buhnova, Grunske, Koziolek, & Meedeniya, 2013). Radliński (2012) conducted a survey to gather expert knowledge about the factors influencing various features of software quality and found that architecture and development platform are the most influential project factor on all software quality features. Software applications can be developed as one-tier (standalone) architecture, two-tier (client-server) architecture or three-tier architecture (multitier). A One-tier application consists of presentation layer, business logic layer and database layer, all located on a single machine. A standalone application is developed such that it can run

on any machine by itself. Most of the mobile applications are standalone applications (Corral, Sillitti, & Succi, 2013). Two-tier architecture software's work through a network connection between the client and the server where client hosts the presentation layer and the server host the business and database layer. Simple web applications development are examples of two-tier architecture. A three-tier application is when each layer of an application is hosted by a different machine. The standalone application is one that is similar to running an application on a personal computer. Typically internet applications employ a three-tier architecture with each providing a particular functionality (Urgaonkar, Shenoy, Chandra, Goyal, & Wood, 2008). By breaking up the software into components on both client and server sides, one can keep the design clean and easily maintainable (Krueger, 2007; Sadikov & Pidkameny, 2014). However, high tired architectures are generally chosen for complex and large projects that involve processing of large volumes of data to increase the performance. But with increased complexity in the software application, the number of defects might also increase. A study conducted by Zimmermann, Premraj, & Zeller (2007) identified that complexity metrics have a significant correlation with pre and post release defects. Hence,

H4: Projects using high-tiered architectural design result lesser quality.

The Team Size of the Project

Many investigations recognize that software team attributes play a critical role in software project success or failure (Espinosa et al., 2007; Šmite, Wohlin, Gorschek, & Feldt, 2010). One of the important team-related attributes that has an enormous impact on project success or failure is team size (Rodriguez, Sicilia, Garcia, & Harrison, 2012). Project team size is considered as an important driver of software development productivity. Project team size impacts schedule decisions which is an important factor in project success (Rodriguez et al., 2012). Empirical results on the impact of team size of productivity are available, but its impact on the quality of software projects are very few. More Empirical research results are necessary to help practitioners to improve the performance of the software development teams (Šmite et al., 2010). Kaur & Sengupta

(2013) state that having a proper team size is essential for a software project and that small group of team results in good communication and tends to be very flexible with a large group of teams. Brooks' law states that merely adding manpower to a late software project gets it delayed (Brooks, 1975). Coordination, management, and information flow become a big task in bigger teams requiring formal process and tools (Mundra, Misra, & Dhawale, 2013). Therefore, larger teams consist of people from different functional expertise and can also lead to communication and collaboration issues, thus reducing the quality. Hence, larger the team size, more could be the error, and less is the quality. Given this, the following hypotheses is proposed.

H5: Projects having larger teams result in less quality software projects.

The Project Size

Previous literature suggests that larger projects have more number of errors resulting in less quality (Martin, Pearson, & Furumo, 2005). Large-scale projects fail 3-5 times more than smaller ones (Charette, 2005). Both static and interactive parts of the project get complicated with the increase of the project size. Size is the single most important factor in quality at CMM level 5 (Ramasubbu, Mithas, Krishnan, & Kemerer, 2008). Larger projects require larger effort and in turn more number of defects, hence less quality compared to projects that require less effort.

H6: Larger projects result in less quality software projects.

2.3.2 Drivers of Effort

The Programming Language Used (generation)

Software programming languages have evolved to a great extent over the years. Compared to the prior generation languages, the later generation languages were developed as an upward trend toward higher abstraction and statement power (Miller, Vandome, & McBrewster, 2010). Later generation languages (4GLs) automate much of the work usually associated with developing software applications (Jiang & Naudé, 2007). One statement of a high-level language code in third generation language is equal to 5 or 10 machine code instructions, while each 4GL statement is equivalent to 30, or even 50, machine code instructions, hence 4GL products can be

developed faster (Schach, 2002). Pendharkar & Rodger (2007) performed an empirical analysis and found that programming language does not have an impact on the development effort. However, in this study, this will be tested with the latest data, hence,

H7: Later generation languages require less development effort.

The Development Methodology Used:

Development methodology aids development, through guidance. It has a set of predefined heuristics which needs to be followed throughout the development process (Sudeikat, Braubach, Pokahr, & Lamersdorf, 2004). The development methods make the software development process streamlined and efficient. Thus, implementing a methodology can impact the estimated effort in the software design. Hence, it is used as one of the drivers of software development effort in empirical models such as Constructive Cost Model (COCOMO) which was developed by Barry. W. Boehm (Pendharkar & Rodger, 2007). Software methodologies can be categorized into heavyweight (focus on detailed documentation, exceptional planning, and extroverted design) and light weight (short cycle times and rely on knowledge within the team) (Khan, Qurashi, & Khan, 2011). According to Khan et al. (2011), development methodologies are aimed at increasing the quality, reduce the development cycle time, provide early risk exposure and maintain good customer relations. Khatibi, Jawawi, & Khatibi (2012) performed an analysis of software projects and found that most of the comparisons showed that using the methodology in software projects increased the amount of development effort considerably. The authors state that methodologies may not always reduce the software effort because lightweight methodologies require detailed documentation that might increase the effort and heavyweight methodologies always require well knowledgeable developers, without which may increase the effort spent on understanding the requirements and developing. The study empirically tests this and also investigates if differences exist between older and newer projects. Therefore, considering the benefits of using development methodologies, hence it is expected that,

H8: Projects using a development methodology require less development effort.

The CASE Tools Used (CASE or otherwise)

CASE tools are expected to impact the software development effort and hence used in models such as COCOMO as one of the drivers of effort (Pendharkar et al., 2005). Computer-Aided Software Engineering (CASE) tools are designed to support software development at all stages of the software development. For instance, the central repository tools provide centralized information for all the developers working in a team. This can help reduce redundancy in work reducing the effort required to build the software application. Few other studies suggest the usage of case tools can increase the effort if the development team is not adaptable and not able to understand the underlying methodologies with the CASE tools (Jiang & Naudé, 2007) Sometimes, forcing the development team to employ tools based on unfamiliar notations may result in spending more time understanding the functionalities increasing the effort. However, we would like to test this with the latest project development data. Hence, the following is proposed,

H9: Project using CASE tools require less development effort.

The Architecture (Standalone or not)

Software architecture acts as a blueprint to the software developers and aid them to easily understand the systems in case of extensions and enhancements etc. As observed earlier, high tired architectures can help improve security, scalability, and availability reusability of the software applications by dividing the systems into subsystems. However, dividing the software into subsystems and components can increase the effort spent on developing, since each subsystem has to be developed separately by experts of the component subsystems. Moreover, high tier architectures are chosen for complex systems with numerous interdependencies among various components across the different tiers (Talwar et al., 2005) requiring more effort than the low tier applications. A change in any one of the tiers in n-tier architecture may require changes to be added to the other tiers to maintain consistency across the tiers (Alves, Valente, & Nunes, 2013). A three-tiered architecture can reduce developmental complexities but replaces them with maintenance difficulties and operations (Elkhatib, Blair, & Surajbali, 2013). Hence, it is postulated that,

H10: Projects using high-tier architectures require more development effort.

Team Size

Human factors are considered as important next to the tools and methods used in software development. Among the human factors team size is an important driver of project success. Previous researches have tried to analyze the relation between team size and project outcomes. Putnam (1978) found that an optimal team size in the development team is important to achieve maximum productivity. Smith, Hale, & Parrish (2001) found that team size does not significantly affect the effort in the development environment. Nevertheless, the study proposes that larger teams require more communication between the team members to exchange views ideas and to know the project progress increasing the project development effort. Although few researchers found that team size does not significantly affect the effort in the development environment (Smith et al., 2001). Other researchers found that team size significantly increases the effort (Pendharkar & Rodger, 2007). However, team size is expected to be an important driver of effort and it is tested with the latest ISBSG data set.

Hence, it is postulated that,

H11: Projects having larger teams require more effort.

Project Size

Software organizations develop different types of projects such as small projects, large projects, new projects and enhancement projects. Large projects generally require more effort since the deliverable is quite large. Moreover, large projects require more number of people to work on it requiring more development effort. In this study, the impact of project size on the effort is tested. Since project size can be a strong determinant in estimating the software development effort. It is postulated that larger projects require more development effort compared to smaller projects. Hence,

H12: Projects that are larger in size require more effort.

The summary of hypotheses and expected relation between the technological and situational drivers and project performance measures: software development quality and effort and is shown in the Table 2.1.

TABLE 2.1: SUMMARY OF HYPOTHESES

H.No	Drivers of Quality and Effort	Hypotheses	Impact on Number of Defects	Impact on Development Effort
H1	Programming Languages	Projects using later generation languages result in high quality software projects.	-	
H2	Development Methodology	Projects using development methodologies result higher quality software projects.	-	
H3	CASE Tools	Projects using CASE tools result in higher quality software projects.	-	
H4	Architecture	Projects using high-tiered architectural design result in less quality of software projects.	+	
H5	Team size	Projects having larger teams result in less quality software projects.	+	
H6	Project Size	Projects that are large in size result in less quality software projects.	+	
H7	Programming Languages	Projects using later generation languages require less development effort.		-
H8	Development Methodology	Projects using a development methodology require less development effort.		-
H9	CASE Tools	Project using case tools require less development effort.		-
H10	Architecture	Projects using high-tiered architectures require more development effort.		+
H11	Team size	Projects having larger teams require more development effort		+
H12	Project Size	Projects that are large in size require more development effort		+

2.4 Research Model

The research model for the study is shown in Figure 2.1.

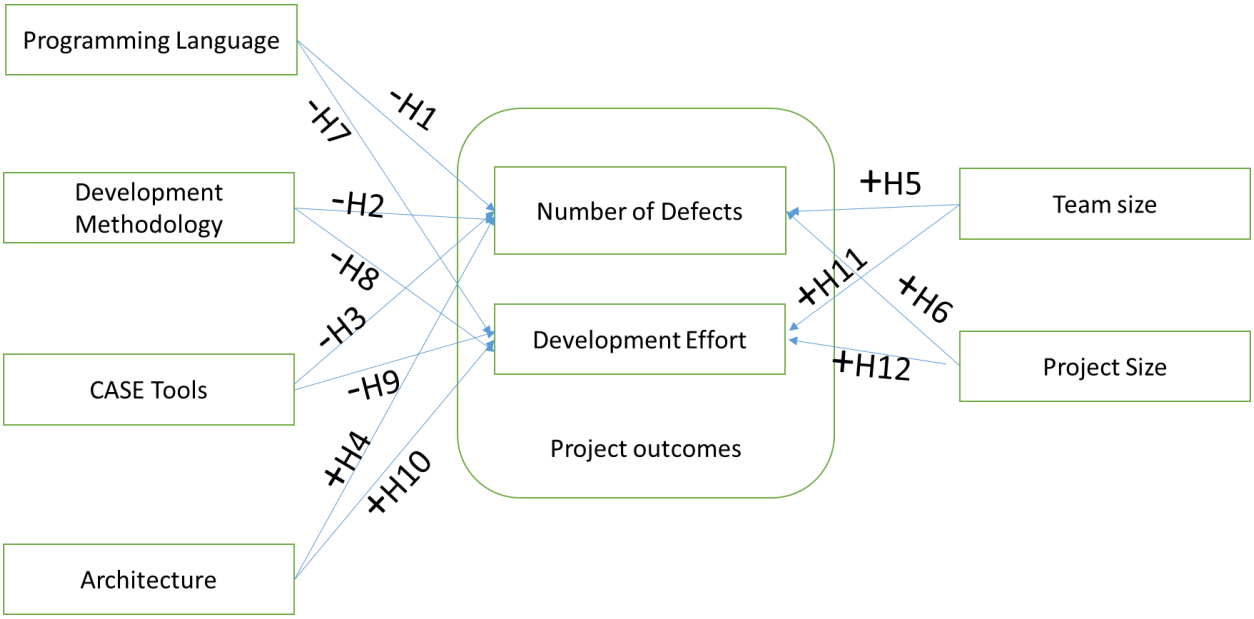


Figure 2.2: The Research Model

2.5 Data

The Release 13 data repository of the International Software Benchmarking Standard Group (ISBSG), is used for analysing 1567 software projects (ISBSG, 2015) for this study. Software project data submission for this repository is voluntary, and some projects do not have data for few data fields that ISBSG offers. The project repository data come from a broad range of project types from different industries, business areas and twenty different countries. The project data was used in various academic studies (Chatzipetrou et al., 2015; Jiang & Naudé, 2007) and many industrial settings for estimation and benchmarking purposes. Among the 6761 software projects from the global project archive, due to missing data for dependent variables, only 1567 qualify as viable projects for the present study. Table 2.2 shows the details of the latent constructs of the research model and the measurement items that are used to measure them for empirical testing of the model for the study.

TABLE 2.2: MEASUREMENT OF VARIABLES

Model Variable Name	**What it measures**	**Type**
Programming Language	Defines the language type used for the project.	2GL/3GL (=1), 4GL (=2), APG (=3) etc.
Development Method	Defines the development methodology used by the development team to build the software.	Yes=1, No=0
CASE tools used	Whether the project used any CASE tool.	Yes=1, No=0
Architecture	Stand-alone or not	Other(multi-tier, client-server)=1, Stand-alone=0
Team Size	The average number of people that worked on the project (calculated from the team sizes per phase).	Number of members in a team
Project Size	For IFPUG counts this is the adjusted functional point size (the functional size is adjusted by a Value Adjustment Factor)	Use Functional Size Measurement Method; For IFPUG counts this is the adjusted size (the functional size is adjusted by a Value Adjustment Factor)
Effort	This value is an estimate of the full development life-cycle effort (normalized work effort).	Number of hours
Quality/Total defects	Total Defects Delivered In the first month of release	Number of defects

Descriptive statistics of the variables used in the study are shown in Table 2.3.

TABLE 2.3: DESCRIPTIVE STATISTICS

Variables Used	**N**	**Minimum**	**Maximum**	**Mean**	**Std. Deviation**
Programming languages	1500	1	3	1.38	0.58
Methodology Used	1323	0	1	0.94	0.23
Case tool used	703	0	1	0.57	0.49
Max Team Size	683	1	309	9.80	14.81
Architecture Used	1258	0	1	0.57	0.49
Project size	1372	4	20000	419.89	1150.96
Effort	1567	8	202386	4799.09	10923.28
Quality/Total Defects	1567	0	2554	13.45	94.73

2.6 Research Method

A Structural equation model (SEM) is developed according to the proposed model and used for answering RQ1 and RQ2. The proposed SEM model is tested using WARPPLS software with Robust Path Analysis algorithm (Kock, 2015). Structural equation models incorporate multiple independent and dependent variables overcoming the problem with regular regression models, and they help analyze the latent constructs that group of observed variables represents (Lowry & Gaskin, 2014; Savalei & Bentler, 2010). For testing, the inner model algorithm was specified as robust path analysis and the outer model was analyzed using the warp 3 algorithm (F. Hair Jr, Sarstedt, Hopkins, & G. Kuppelwieser, 2014). The inner model is the structural model that displays the relationships between constructs or the latent variables, and the outer model is the measurement model that is used to evaluate the relationships between the indicator variables and their corresponding constructs (F. Hair Jr et al., 2014). The research models were tested for the overall validity and the path validity between the dependent variables (software effort and number of defects). The path coefficients were then analyzed to test the relationship between the dependent and independent variables.

2.7 Results

The analysis of the results from the SEM was performed in two stages. In the first stage, the overall fit of the model was tested using the model fit indices suggested by (Kock, 2015) and the measurement model was validated using the convergent and discriminant validity. In stage two, the structural model was analyzed to test the hypotheses and the path coefficients for the relationships between the exogenous and the endogenous variables.

The model fit indices from the SEM analysis for the research model is shown in Table 2.4. According to Kock (2015) it is recommended that the P-values for the Average path coefficient (APC), Average R-squared (ARS) and Average Adjusted R-squared (AARS) all be equal to or lower than 0.05; that is, significant at the 0.05 level. All the three models satisfy this condition. The Tenenhaus GoF index is a measure of a model's explanatory power. The results show that for

all the three models the GOF value is greater than 0.5 indicating that all the three models have high explanatory power. All the fit indices satisfy the acceptable limits, hence the research model is identified as stable.

TABLE 2.4: MODEL FIT INDICES

Model fit indices	Research model	Recommended value
Average path coefficient (APC)	0.128, P<0.001	p<0.05
Average R-squared (ARS)	0.268, P<0.001	p<0.05
Average adjusted R-squared (AARS)	0.265, P<0.001	p<0.05
Average block VIF (AVIF)	1.09	acceptable if <= 5, ideally <= 3.3
Average full collinearity VIF (AFVIF)	1.146	acceptable if <= 5, ideally <= 3.3
Tenenhaus GoF (GoF)	0.518	small >= 0.1, medium >= 0.25, large >= 0.36
Sympson's paradox ratio (SPR)	0.692	acceptable if >= 0.7, ideally = 1
R-squared contribution ratio (RSCR)	0.988	acceptable if >= 0.9, ideally = 1
Statistical suppression ratio (SSR)	0.917	acceptable if >= 0.7
Nonlinear bivariate causality direction ratio (NLBCDR)	0.833	acceptable if >= 0.7

2.7.1 Measurement Model

Convergent validity, discriminant validity are analyzed to test the validity of the measurement model. The convergent validity tests the pattern of loadings of the measurement items into the latent variables and discriminant validity measures the quality of the measurement instrument. Convergent validity was tested by performing the confirmatory factor analysis with the factor loadings and cross-loadings. For convergent validity, the factor loadings have to be greater than 0.5 (Hair et al., 1998). Results from the confirmatory factor analysis for the "all projects model" shown in Table 2.5 indicate that all the observed variable load well into the latent constructs without any significant cross-loadings. Discriminant validity was tested by comparing

the average variance extracted to the correlations among the latent variables. For discriminant validity, the square root of average variance extracted should be greater than the correlations among the latent constructs (Fornell & Larcker, 1981). Correlations between latent constructs and average variance extracted (AVE) in the diagonal shown in Table 2.6. The table shows that the average variance extracted for the constructs is greater than the correlations between the constructs thus showing evidence of discriminant validity. The composite reliability tests the reliability of a latent variable which is similar to the Cronbach alpha. The VIF'S for the three models were also tested to check for multicollinearity issues between the latent variables which indicate that two latent variables measure the same thing (Hair, Black, and Anderson, 1998). Results show that there are no multicollinearity issues between the latent variables. Thus, the model fit indices and the measurement model validates the stability of the research model of the study. Further section provides results for the structural model.

TABLE 2.5: FACTOR LOADINGS AND CROSS LOADINGS FOR THE RESEARCH MODEL

	Case tools used	**Number of defects**	**Architecture**	**Methodology used**	**Effort**	**Languages used**	**Team Size**	**Project size**
Case tools used	**0.94**	-0.155	0.09	0.272	-0.025	-0.063	-0.021	-0.077
Number of defects	-0.147	**0.893**	0.007	-0.273	0.167	0.025	0.023	0.279
Architecture	0.094	0.008	**0.978**	0.107	0.027	-0.135	0.039	-0.049
Methodology used	0.258	-0.273	0.097	**0.891**	-0.087	-0.035	0.01	-0.215
Effort	-0.024	0.169	0.025	-0.088	**0.899**	-0.04	0.276	0.276
Languages used	-0.066	0.027	-0.136	-0.039	-0.043	**0.985**	-0.042	0.028
Team Size	-0.021	0.025	0.038	0.011	0.291	-0.041	**0.949**	0.1
Project size	-0.073	0.277	-0.045	-0.214	0.272	0.025	0.094	**0.887**

TABLE 2.6: CORRELATIONS AND SQUARE ROOT OF AVERAGE VARIANCE EXTRACTED FOR RESEARCH MODEL

	Case tools used	Number of defects	Architecture	Methodology used	Effort	Languages used	Team Size	Project size
Case tools used	1							
Number of defects	-0.165**	1						
Architecture	0.096**	0.008	1					
Methodology used	0.289**	-0.306**	0.109**	1				
Effort	-0.027	0.187**	0.028	-0.098**	1			
Languages used	-0.067**	0.028	-0.138**	-0.039	-0.044**	1		
Team Size	-0.022	0.026	0.04	0.011	0.306**	-0.043	1	
Project size	-0.082**	0.313**	-0.05**	-0.241**	0.307**	0.028	0.106**	1

*Square root of Average variance extracted on the diagonal; ** p<0.05*

2.7.2 Structural Model

The path coefficients and amount of variance extracted were analyzed to measure the explanatory power of the structural model. The path coefficient results from the SEM analysis can be seen in Table 2.7. In the research model the independent variables explained 20 % of the variance in the number of defects and 33 % variance in software development effort. Results suggest that usage of technological factors such as development methodology (β= -0.209; $p < 0.001$), high tiered architecture (client-server or multitier) ($\beta = 0.039$; $p < 0.061$) and case tools (β = -0.091; $p < 0.001$) have a significant impact on the number of defects while the type of programming language used (β= -0.023; $p < 0.176$) did not have any impact on the number of defects. This suggests that technical factors play a critical role in determining the quality of the software projects. Although results suggested that using higher generation programming languages

can reduce ((β= -0.023^{ns}) the number of defects the result was not significant. Thus we could not find support for hypotheses H1. However, results suggest that using a suitable development methodology (β= -0.209), and CASE tools (β = -0.091) can help reduce the number of defects in a software development project, thus delivering the project with good quality supporting hypotheses H2 and H3. Using high-tiered architecture in the software design can reduce the quality due to the increased number of defects supporting hypotheses H4. Situational factors team size (β = -0.032; p = 0.099) and project size (β = 0.322; p < 0.001) also had significant effect on the quality of software development. It is found that larger team size is associated with less number of defects, thus, not supporting H5. Larger project size is associated with more number of defects supporting H6. Thus, among the six hypotheses related to the number of defects, the study finds support for four hypotheses (H2, H3, H4 and H6). Summary of results of the hypotheses can be seen in Table 2.8.

TABLE 2.7: PATH COEFFICIENTS FROM SEM ANALYSIS

Path Effects	**Path coefficients**
Programming language --> Number of defects	-0.023
Development Methodology --> Number of defects	-0.209***
CASE Tools --> Number of defects	-0.091***
Architecture --> Number of defects	0.039*
Team size --> Number of defects	-0.032*
Project size--> Number of defects	0.322***
Programming language --> Effort	-0.053**
Development Methodology --> Effort	0.005
CASE Tools --> Effort	0.022
Architecture --> Effort	0.009
Team size--> Effort	0.291***
Project size --> Effort	0.443***
R- square	0.27
****p<0.001; **p<0.05 ;*p<0.1*	

TABLE 2.8: RESULTS OF HYPOTHESES

	Drivers of Project Outcomes	Expected Impact on Number of Defects	Expected Impact on Development Effort	All Projects
H1	Programming Languages	-		non-significant
H2	Development Methodology	-		supported
H3	CASE Tools	-		supported
H4	Architecture	+		supported
H5	Team size	+		Not supported
H6	Project size	+		supported
H7	Programming Languages		-	supported
H8	Development Methodology		-	non-significant
H9	CASE Tools		-	non-significant
H10	Architecture		+	non-significant
H11	Team size		+	supported
H12	Project size		+	supported

Results with respect to the development effort as dependent variable show that usage of higher generation programming languages ($\beta = -0.053$; $p < 0.017$) can reduce development effort required for the software project, thus supporting hypotheses H7. However, usage of other technical factors such as development methodology ($\beta = 0.005$; $p < 0.428$), CASE tools ($\beta = 0.022$; $p < 0.196$) and type of architecture ($\beta = 0.009$; $p < 0.359$) did not have any significant impact on the development effort, thus not finding support for hypotheses H8, H9 and H10. This suggests that technical factors do not play a significant role in determining the effort required for the software development. Situational factors such as team size ($\beta = 0.322$; $p < 0.001$) and project size ($\beta = 0.322$; $p < 0.001$) has strong significant effect on the development effort indicating that larger teams and larger projects require more effort for software development supporting hypotheses H11 and H12. Thus among the six hypotheses with respect to the development effort, the results support for three hypotheses (H7, H11, H12).

2.8 Discussion

Results from the fit indices show that the SEM model has good explanatory power in determining the technological determinants of software development projects given the R-squared value of 0.27. The path coefficients show that technological and situational factors play a critical role in determining the software development quality whereas the technical factors do not play a very critical role in determining the software development effort unlike the situational factors.

It is observed that usage of higher generation of programming languages did not have any impact on the quality of the software development thus, not supporting H1, but using higher generation programming languages can help reduce the development effort required for software development, thus supporting H7. This suggests that usage of higher generation programing languages (4GL, APG) is beneficial for reducing the development effort as they come with high amount of automation and advanced features and capabilities compared to the older generation languages(2GL, 3GL) (Jiang & Naudé, 2007; Miller et al., 2010). This result was in contrary to the findings provided by Pendharkar & Rodger (2007) that programming language did not have any impact on the development effort. The study results also support various other authors who stated that later generation languages come with high abstraction and automation which is the main reason for the reduced development effort that is required. The results are in contrast to the notion that later generation languages can introduce complexity and lead to reduced positive outcomes (Becerril, 2011; Chatzipetrou et al., 2015; Kumaresh & Baskaran, 2010). Hence, the study suggests the managers to use later generation languages. While it is true that introduction of new programming languages can initially be problematic if the developers are not comfortable in using them. These complexities can be reduced by providing required training to the development team when new programming languages are introduced.

It is also observed that usage of development methodologies helps reduce the number of defects increasing the quality of the software product suggesting that a better process results in better products (Baggen et al., 2012), thus supporting the Hypotheses H2. But support for hypotheses H8 with states that implementing development methodologies can reduce the

development effort was not found. This is possible because the projects used in the study use various traditional and latest methodologies. While the traditional methodologies focus more on documentation like the waterfall development , the later generation methodologies like agile focus more on quickly meeting the deadlines and delivering the work with the agreed upon SLA'S rather than documentation which may need less effort hours (Erickson et al., 2005). This could be a possible reason for the non-supporting result of the effect of development methodology on development effort. To understand this effect, further investigation about the effect of different types of development methodologies on the development effort may be required.

Results also suggested that usage of case tools can enhance the quality of the software development product, thus supporting hypotheses H3 but cannot reduce the development effort thus not supporting hypotheses H9.Although CASE tools help automate the development process to an extent and make the maintenance activities easier they do not play a critical role in reducing the effort. The results partially support the statement given by Coronel & Morris (2016) that CASE tools are intended to improve quality and reduce time of the software development.

As expected, it is found that usage of high tired architectures results in reduced quality of the software projects supporting hypotheses H4. The main reason for this is that the high tired architecture are generally chosen for complex projects to simplify the design. However, this simplification is not successful in improving the quality. Hence, it is suggested that high tired architectures are to be chosen only when it is very much required for the project. Usage of high tired-architectures did not have any impact on the development effort, thus not supporting hypotheses H10. To sum up, it is observed that, technical factors had a significant impact on the quality but do not have a prominent and significant effect on the software development effort.

Unlike the technical factors, the situational factors are found to have a significant impact on the software development project outcomes supporting H6, H11 and H12. Results show that lager team sizes can hamper the project outcomes requiring large development effort. This suggests that it is important to be cautious about the development team sizes. The software development managers should note that large team sizes does not always result in positive

outcomes unless the team dynamics are appropriately managed in the form of improved coordination, communication, social dynamics etc. (Mundra et al., 2013). This also suggests that having small team sizes is always beneficial for the software development (Kaur & Sengupta, 2013). As expected large project sizes are associated with more effort and more number of defects. This suggests that there is a need to manage the bigger projects with much more care in order to make them successful.

The study provide important implications for the managers of the software development teams in terms of the types of technologies which can enhance the software development project performance. In order to have better quality software products it is very important to focus on the type of technologies that are used in software development in addition to the team and organizational factors.

While the study analyzes the impact of technological factors on project performance using a large number of projects, the results need to be analyzed with care since differences could exist between new and enhanced projects. Therefore, the future study for this research is to explore the differences that exist between new and enhanced projects.

2.9 Conclusion

The study aims at identifying and testing the impact of technological determinants on software development quality and development effort. Results from the empirical tests suggest that technological factors significantly impact the quality of the software products but not the software development effort. It is found that usage of development methodologies and CASE tools can help improve the quality of software development, whereas usage of high-tired architectures can reduce the quality of software development. Technical factors did not significantly impact the software development effort except for the usage of higher generation programming languages. Higher generation programming languages can reduce the development effort required due to their enhanced features and high extent of automation features. Thus, it can be understood that technical factors play an important role in software development. It is found

that smaller team sizes are always beneficial for the software development since these projects are found to have better quality along with reduced effort. It is also a good practice to carefully monitor the large projects as they are more prone to failures in terms of quality and development effort required.

2.10 References

Agrawal, M., & Chari, K. (2007a). Software effort, quality, and cycle time: A study of CMM level 5 projects. *IEEE Transactions on Software Engineering, 33*(3), 145-156.

Al-Ahmad, W., Al-Fagih, K., Khanfar, K., Alsamara, K., Abuleil, S., & Abu-Salem, H. (2009). A taxonomy of an IT project failure: Root causes. *International Management Review, 5*(1), 93.

Aleti, A., Buhnova, B., Grunske, L., Koziolek, A., & Meedeniya, I. (2013). Software architecture optimization methods: A systematic literature review. *Software Engineering, IEEE Transactions On, 39*(5), 658-683.

Alves, R., Valente, P., & Nunes, N. J. (2013). Improving software effort estimation with human-centric models: A comparison of UCP and iUCP accuracy. Paper presented at the *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 287-296.

Awad, M. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia,*

Baggen, R., Correia, J. P., Schill, K., & Visser, J. (2012). Standardized code quality benchmarking for improving software maintainability. *Software Quality Journal, 20*(2), 287-307.

Basha, S., & Ponnurangam, D. (2010). Analysis of empirical software effort estimation models. *arXiv Preprint arXiv:1004.1239,*

Basili, V. R., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., Trendowicz, A. (2013). Linking software development and business strategy through measurement. *arXiv Preprint arXiv:1311.6224,*

Bayraktaroglu, A., Calisir, F., & Gumussoy, C. (2009). Usability and functionality: A comparison of project managers' and potential users' evaluations. Paper presented at the *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference On*, 2019-2023.

Becerril, M. G. G. (2011). Reengineering a cloud computing platform using model based technologies.

Brooks, F. P. (1975). *The mythical man-month* Addison-Wesley Reading, MA.

Chappell, D. (2012). Business value of software quality. Paper presented at the *North America Quest Conference& Expo 2012–Quality Engineered Software &Testing,*

Charette, R. N. (2005). Why software fails. *IEEE Spectrum*, *42*(9), 36.

Chatzipetrou, P., Papatheocharous, E., Angelis, L., & Andreou, A. S. (2015). A multivariate statistical framework for the analysis of software effort phase distribution. *Information and Software Technology, 59*, 149-169.

Clarke, P., & O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, *54*(5), 433-447.

Coronel, C., & Morris, S. (2016). *Database systems: Design, implementation, & management* Cengage Learning.

Corral, L., Sillitti, A., & Succi, G. (2013). Software development processes for mobile systems: Is agile really taking over the business? Paper presented at the *Engineering of Mobile-Enabled Systems (MOBS), 2013 1st International Workshop on The*, 19-24.

Dwivedi, Y. K., Ravichandran, K., Williams, M. D., Miller, S., Lal, B., Antony, G. V., & Kartik, M. (2013). IS/IT project failures: A review of the extant literature for deriving a taxonomy of failure factors. *Grand successes and failures in IT. public and private sectors* (pp. 73-88) Springer.

El Emam, K., & Koru, A. G. (2008). A replicated survey of IT software project failures. *Software, IEEE*, *25*(5), 84-90.

Elkhatib, Y., Blair, G. S., & Surajbali, B. (2013). Experiences of using a hybrid cloud to construct an environmental virtual observatory. Paper presented at the *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*, 13-18.

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management, 16*(4), 88.

Espinosa, J. A., Slaughter, S. A., Kraut, R. E., & Herbsleb, J. D. (2007). Familiarity, complexity, and team performance in geographically distributed software development. *Organization Science, 18*(4), 613-630.

F. Hair Jr, J., Sarstedt, M., Hopkins, L., & G. Kuppelwieser, V. (2014). Partial least squares structural equation modeling (PLS-SEM) an emerging tool in business research. *European Business Review, 26*(2), 106-121.

Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research*, , 39-50.

Fuggetta, A., & Di Nitto, E. (2014). Software process. Paper presented at the *Proceedings of the on Future of Software Engineering*, 1-12.

Garcia-Diaz, N., Lopez-Martin, C., & Chavoya, A. (2013). A comparative study of two fuzzy logic models for software development effort estimation. *Procedia Technology, 7*, 305-314.

Gillies, A. (2011). *Software quality: Theory and management* Lulu. com.

Gorla, N., & Lin, S. (2010). Determinants of software quality: A survey of information systems project managers. *Information and Software Technology, 52*(6), 602-610.

Hair, J. F., Anderson, R. E., Tatham, R. L., & William, C. (1998). Black (1998), multivariate data analysis.

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

Huang, S., & Chiu, N. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and Software Technology, 48*(11), 1034-1045.

Jiang, Z., & Naudé, P. (2007). An examination of the factors influencing software development effort. *International Journal of Computer Information and Systems Science and Engineering, 1*(3), 182-191.

Jodpimai, P., Sophatsathit, P., & Lursinsap, C. (2010). Estimating software effort with minimum features using neural functional approximation. Paper presented at the *2010 International Conference on Computational Science and its Applications*, 266-273.

Jones, C. (2006). Social and technical reasons for software project failures. *STSC Cr0ssTalk June*,

Jørgensen, M. (2010). Identification of more risks can lead to increased over-optimism of and over-confidence in software development effort estimates. *Information and Software Technology, 52*(5), 506-516.

Jørgensen, M. (2014). Failure factors of small software projects at a global outsourcing marketplace. *Journal of Systems and Software, 92*, 157-169.

Jorgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *Software Engineering, IEEE Transactions On, 33*(1), 33-53.

Jørgensen, M., & Sjøberg, D. I. (2004). The impact of customer expectation on software development effort estimates. *International Journal of Project Management, 22*(4), 317-325.

Juristo, N., Moreno, A. M., & Sanchez-Segura, M. (2007). Analysing the impact of usability on software design. *Journal of Systems and Software, 80*(9), 1506-1516.

Kaur, R., & Sengupta, J. (2013). Software process models and analysis on failure of software development projects. *arXiv Preprint arXiv:1306.1068,*

Khan, A. I., Qurashi, R. J., & Khan, U. A. (2011). A comprehensive study of commonly practiced heavy and light weight software methodologies. *arXiv Preprint arXiv:1111.3001,*

Khatibi, V., Jawawi, D. N., & Khatibi, E. (2012). Investigating the effect of using methodology on development effort in software projects.

Kock, N. (2015). WarpPLS 5.0 user manual. *Laredo, TX: ScriptWarp Systems,*

Krishnan, M. S., Kriebel, C. H., Kekre, S., & Mukhopadhyay, T. (2000). An empirical analysis of productivity and quality in software products. *Management Science, 46*(6), 745-759.

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software,* (6), 18-21.

Krueger, C. W. (2007). The 3-tiered methodology: Pragmatic insights from new generation software product lines. Paper presented at the *Software Product Line Conference, 2007. SPLC 2007. 11th International,* 97-106.

Kumaresh, S., & Baskaran, R. (2010). Defect analysis and prevention for software process quality improvement. *International Journal of Computer Applications, 8*(7)

Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived causes of software project failures–An analysis of their relationships. *Information and Software Technology, 56*(6), 623-643.

Lending, D., & Chervany, N. L. (1998). The use of CASE tools. Paper presented at the *Proceedings of the 1998 ACM SIGCPR Conference on Computer Personnel Research,* 49-58.

Lowry, P. B., & Gaskin, J. (2014). Partial least squares (PLS) structural equation modeling (SEM) for building and testing behavioral causal theory: When to choose it and how to use it. *Professional Communication, IEEE Transactions On, 57*(2), 123-146.

Luftmann, J., & Kempaiah, R. (2008). Key issues for IT executives 2007. *MIS Quarterly Executive, 7*(2)

Marasco, J., & Ravenflow, C. (2006). Software development productivity and project success rates: Are we attacking the right problem. *The Rational Edge, February, 15*

Martin, N. L., Pearson, J. M., & Furumo, K. A. (2005). IS project management: Size, complexity, practices and the project management office. Paper presented at the *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference On,* 234b-234b.

Miller, F. P., Vandome, A. F., & McBrewster, J. (2010). Fourth-generation programming language: Programming language, history of computer science, third-generation programming language, fifth-generation programming... problem solving, systems engineering.

Mundra, A., Misra, S., & Dhawale, C. A. (2013). Practical scrum-scrum team: Way to produce successful and quality software. Paper presented at the *Computational Science and its Applications (ICCSA), 2013 13th International Conference On,* 119-123.

Nasir, M., & Sahibuddin, S. (2011). Addressing a critical success factor for software projects: A multi-round delphi study of TSP. *International Journal of the Physical Sciences, 6*(5), 1213-1232.

Nelson, R. R., & Morris, M. G. (2014). IT project estimation: Contemporary practices and management guidelines. *MIS Quarterly Executive, 13*(1)

Palvia, S. C., Sharma, R. S., & Conrath, D. W. (2001). A socio-technical framework for quality assessment of computer information systems. *Industrial Management & Data Systems, 101*(5), 237-251.

Pendharkar, P. C., & Rodger, J. A. (2007). An empirical study of the impact of team size on software development effort. *Information Technology and Management, 8*(4), 253-262.

Pendharkar, P. C., Subramanian, G. H., & Rodger, J. A. (2005). A probabilistic model for predicting software development effort. *Software Engineering, IEEE Transactions On, 31*(7), 615-624.

Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering, 4*(4), 345.

Radliński, Ł. (2012). Towards expert-based modelling of integrated software quality. *Journal of Theoretical and Applied Computer Science, 6*(2), 13-26.

Ramasubbu, N., Mithas, S., Krishnan, M. S., & Kemerer, C. F. (2008). Work dispersion, process-based learning, and offshore software development performance. *MIS Quarterly,* , 437-458.

Ray, B., Posnett, D., Filkov, V., & Devanbu, P. (2014). A large scale study of programming languages and code quality in github. Paper presented at the *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering,* 155-165.

Rodriguez, D., Sicilia, M., Garcia, E., & Harrison, R. (2012). Empirical findings on team size and productivity in software development. *Journal of Systems and Software, 85*(3), 562-570.

Sadikov, V., & Pidkameny, W. (2014). Complete separation of the 3 tiers-divide and conquer. *arXiv Preprint arXiv:1405.1618,*

Samadhiya, D., Wang, S., & Chen, D. (2010). Quality models: Role and value in software engineering. Paper presented at the *Software Technology and Engineering (ICSTE), 2010 2nd International Conference On, , 1* V1-320-V1-324.

Sandhu, G. S., & Salaria, D. S. (2014). A bayesian network model of the particle swarm optimization for software effort estimation. *International Journal of Computer Applications, 96*(4)

Savalei, V., & Bentler, P. M. (2010). Structural equation modeling. *Corsini Encyclopedia of Psychology,*

Schach, S. R. (2002). *Object-oriented and classical software engineering* McGraw-Hill New York.

Schmidt, D. C. (2006). Model-driven engineering. *Computer-Ieee Computer Society-, 39*(2), 25.

Šmite, D., Wohlin, C., Gorschek, T., & Feldt, R. (2010). Empirical evidence in global software engineering: A systematic review. *Empirical Software Engineering, 15*(1), 91-118.

Smith, R. K., Hale, J. E., & Parrish, A. S. (2001). An empirical study using task assignment patterns to improve the accuracy of software effort estimation. *IEEE Transactions on Software Engineering, 27*(3), 264-271.

Staron, M. (2012). Critical role of measures in decision processes: Managerial and technical measures in the context of large software development organizations. *Information and Software Technology, 54*(8), 887-899.

Sudeikat, J., Braubach, L., Pokahr, A., & Lamersdorf, W. (2004). Evaluation of agent–oriented software methodologies–examination of the gap between modeling and platform. *Agent-oriented software engineering V* (pp. 126-141) Springer.

Talwar, V., Wu, Q., Pu, C., Yan, W., Jung, G., & Milojicic, D. (2005). Comparison of approaches to service deployment. Paper presented at the *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference On,* 543-552.

Tierno, I. A., & Nunes, D. J. (2012). Assessment of automatically built bayesian networks in software effort prediction. Paper presented at the *CIbSE,* 196-209.

Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., & Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS), 3*(1), 1.

Verner, J. M., Babar, M. A., Cerpa, N., Hall, T., & Beecham, S. (2014). Factors that motivate software engineering teams: A four country empirical study. *Journal of Systems and Software, 92,* 115-127.

Zimmermann, T., Premraj, R., & Zeller, A. (2007). Predicting defects for eclipse. Paper presented at the *Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop On*, 9-9.

Chapter 3: Co-Diffusion Trends in Software Sourcing Arrangements

3.1 Introduction

Information Technology assets play an important role in the success of organizations. Historically, organizations developed their software (Scott, 2007). But the complexity of the Information Technology (IT) development and its huge demand led the core IT and non-IT firms to obtain their IT assets through various forms of software sourcing arrangements. Software development ecosystem has greatly evolved over the years with different forms of software sourcing options such as in-housing, on-shoring, offshoring, outsourcing, etc. In the recent years, latest forms of software sourcing have also evolved such as crowd sourcing, inner sourcing, open sourcing, and cloud sourcing. The emergence of various trends of software sourcing with options to buy or build has attracted many organizations to adopt them widely in the past two decades. Software sourcing is considered as a cost-saving option for obtaining the valuable IT resources for organizations where software development is considered as a non-core activity. Software sourcing is also a viable option for the organizations to take advantage of cost reduction, follow the sun strategy, obtaining the world-class expertise and the ability to focus on the core competencies, given the complexity of the software systems (Ågerfalk, Fitzgerald, & Stol, 2015).

Since, there are various alternatives for obtaining enterprise software, choosing and adopting the right sourcing option is a complex task for the IT/IS (/Information Systems) management. IT/ IS management consider the advantages and disadvantages about each sourcing option to choose the sourcing option that best suits for their business. Software sourcing can be broadly classified into four types based on geography (on-shoring and offshoring) and perceived externalization (in-house and outsource) (Contractor, Kumar, Kundu, & Pedersen, 2010; Kedia & Mukherjee, 2009) as shown in Figure 3.1. If the IT project team completely develops the software project within an organization and within the country, it is termed as complete in-housing. When the firms face challenges in the form of increased customer demand and a strong requirement to focus on the core competencies, firms will look out for externalization advantages from external suppliers residing the same country which is termed as domestic outsourcing. However, if the firms

would consider taking advantage of both the externalization as well as the world class expertise around the world to meet their customer demands, offshore outsourcing turns out to be a suitable option. But, if the firms consider the risks associated with knowledge dissipation when associated with external suppliers as a priority but have the capacity to set up foreign subsidiaries to take advantage of talented knowledge pool they can choose captive offshoring. Therefore, choice of software sourcing decisions is based on the insights that the managers have about the organizations' structure, strengths, processes, interdependencies, etc. (Schneider & Sunyaev, 2016).

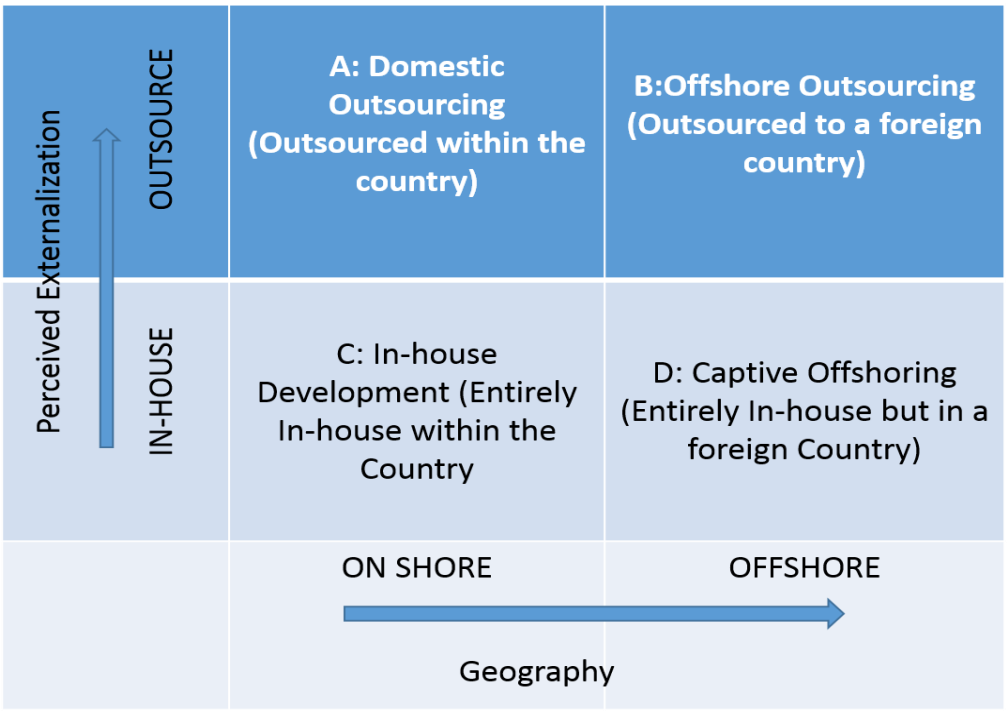


Figure 3.1: Software Sourcing Models

Albeit the existence of four different software sourcing models, offshore outsourcing has gained wide popularity. Offshore outsourcing has been adopted widely by organizations since its evolution between the year 1995 and 2000 with growth and technological improvements in information and communication technology (Palugod & Palugod, 2011). Research about software sourcing has evolved and expanded to a great extent over the past two decades significantly.

Researchers have explored about the merits, demerits, consequences of adoption, analyzed case studies of software sourcing adoption, factors affecting, decision making about the software sourcing options etc. to understand more about the in-housing, outsourcing, offshoring, and on-shoring. Among the many existing studies about software sourcing, most of them find their study object, in the organization as whole and few others, find the project as their study object while a third focus area is about the consequences of adopting software sourcing in certain locations (Moe, Šmite, Hanssen, & Barney, 2014).

Yadav & Gupta (2008) performed an exploratory analysis of content from various research papers published in eight notable journals about outsourcing. While the authors note that IS outsourcing is a multifaceted subject and has been a favorite topic of research with increasing number of studies, there is an apparent need to move beyond snapshot studies and to incorporate temporal effects to answer questions, like How does outsourcing process change over time?. Given the vast existing literature about different modes of software sourcing, it is very well evident that researchers performed many snapshot studies to understand the adoption of software sourcing at the organization level and the project level at large. But, the study about the diffusion of the software sourcing which analyses the spread of software sourcing phenomena over time into various countries has been ignored (Lewin & Volberda, 2011). Lewin & Volberda, (2011) note that there is no international business theory which explains about how offshoring develops over time as they do. Alsudairi & Dwivedi (2010) suggest that diffusion of software sourcing is one of the future trends that needs to be focused by the IS researchers since it can contribute to the evolution and development of the software sourcing options. Aspray, Mayadas, & Vardi (2009) note that software sourcing which is currently considered as a trade of comparative advantage may become less favorable and can lead to a conflict of interest in the current adoptees. To determine such inconsistencies, predictive models over time assist in understanding the future and current trends in innovation adoption. Diffusion studies and innovation diffusion models help understand the channels through with an innovation spreads over time. They can also assist in understanding the past, present and the future trends in the diffusion of an innovation.

Despite the call for diffusion studies in IS literature, there are very few existing studies that study the temporal/time-based spread of the software sourcing phenomenon. Few notable studies that used the innovation diffusion studies to understand the temporal spread and the factors effecting the temporal spread of software sourcing are performed by Chakrabarty (2006); Chaudhury & Bharati, (2008); Hu, Saunders, & Gebelt (1997); Loh & Venkatraman (1992); Mann, Kauffman, Han, & Nault (2011). The authors identified information systems outsourcing as an innovation and adopted the innovation diffusion models to understand the influence factors that impact the spread of innovation over time. However, further expanding the diffusion studies in software sourcing can assist in predicting the future trends of the diffusive adoption of the software sourcing arrangements. While the researchers analyzed the diffusion of software sourcing over time, it is to be noted that diffusion effect is not the sole reason for the dissemination of an innovation as new products seldom are unaffected by the other existing or new products. It would be considered as a limitation if it is assumed that innovations diffuse on their own because products have diffusive interactions with the other existing or new products. The introduction of a new innovation not only delivers the envisioned benefits but might affect the diffusion of another existing or a new innovation due to interactions between them called as co-diffusion effects. Bucklin & Sengupta (1993) define co-diffusion as "the interaction between the demands of innovations that have separate adoption tracks and sometimes co-diffusion effects between innovations are stronger than innovation effects". Co-diffusion is another dimension of diffusion that has been underexplored in the IS research. Since software sourcing is identified as an innovation we expect that the adoption of innovative sourcing strategies such as offshoring and outsourcing can have co-diffusive effects on the existing sourcing phenomena such as on-shoring and in-housing. Through this research, the co-diffusive effects of the software sourcing alignments are analyzed.

Co-diffusion effects exist between two related innovations that have separate adoption tracks. The current study aims to analyze the co-diffusion effects between software sourcing options that can be classified based on the geography: offshoring and on-shoring and that can be

classified based on perceived externalization: in-housing and outsourcing. Thus, the study accomplishes two research objectives. The first research objective of the study is to identify whether co-diffusion effects exist between the software sourcing arrangements and to determine if diffusion effects are stronger than the co-diffusion effects over time. The second objective of the study is to identify the nature of co-diffusion effects between the software sourcing phenomena: 1) Off-shoring and on shoring and 2) In housing and Outsourcing. Till date, we did not locate any research paper that analyze the co-diffusive effects between the software sourcing phenomena. Hence, this co-diffusion research will contribute to the overall software sourcing literature by exploring the temporal diffusion of the software sourcing arrangements.

3.2 Theory

The current study leverages on the widely accepted Innovation Diffusion Theory by Rogers (Rogers, 2010), Bass Diffusion model (Bass, 1969) and innovation interrelationships identified by Peterson and Mahajan (Mahajan & Peterson, 1985; Peterson & Mahajan, 1978).

According to Rogers (2010), diffusion is a kind of social change and is defined as "the process by which alteration occurs in the structure and function of the social system due to the spread of both planned and spontaneous new ideas". Rogers identifies that there are four key elements in the diffusion process of innovation: 1) Innovation 2) Communication channels 3) Time 4) Social System. Rogers also notes that an innovation can be any new idea, practice or object perceived as new by an individual or any unit of adoption; examples are political philosophy such as Marxism, a policy such as municipal nonsmoking ordinance, etc. In view of this, different forms of software sourcing arrangements such as in-housing, outsourcing, offshoring and on-shoring are identified as innovations. According to the innovation diffusion theory, the rate of adoption of an innovation depends on the characteristics of the innovation as perceived by the members of a social system through the communication channels. The communication channel is the means by which message about the innovation characteristics spread from one individual to the other. Communication channels can influence the decision to adopt or reject a new idea. Communication

channels can be mass media channels (external influence channels) or interpersonal communication/word of mouth (internal influence channels) among peers. The third element is the time taken to make a decision to adopt or reject the new idea and the social system is the set of interrelated units that are engaged in making the decision about adopting a new innovation.

3.2.1Fundamental Diffusion Model

The time-based diffusion pattern of an innovation when plotted, generally takes the form of an S-shaped curve (Rogers, 2010). However, the exact shape including the slope and the shape may differ, where a steep slope indicates a rapid diffusion and a gradual slope represents slow diffusion. The purpose of the diffusion model is to depict successive increases in the number of adopters and predict the continued development of a diffusion process that is already in progress (Mahajan, Muller, & Bass, 1991). The rate of diffusion for the fundamental diffusion model can be represented using the mathematical equation (Mahajan & Peterson, 1985; Peterson & Mahajan, 1978; Teng, Grover, & Guttler, 2002) as shown in Equation 1 below.

$$\frac{dN(t)}{dt} = g(t)(m - N(t)) \qquad \text{Equation (1)}$$

Where,

N (t) is the number of adopters that have adopted the innovation at time (t); (t $(0 \leq N(t) \leq m)$

$\frac{dN(t)}{dt}$ is the rate of adoption of the innovation

$'m'$ is the total number of potential adopters in the social system; saturation level

$g(t)$ is the coefficient of diffusion that indicates the communication channels through which the innovation diffusion occurs

‘t' is time of adoption

3.2.2 Mixed Influence Model

In the year 1969, Frank Bass developed a mathematical model that can predict the rate of adoption of a new innovation called as the Bass Diffusion model. It is a predictive model that seeks

to forecast how many adoptions of a new product will occur at future time periods. It is an aggregated forecast of the total number of adopters in each time period (Bass, 1969; Teng et al., 2002). It leverages on the Rogers diffusion theory that the potential adopters of an innovation are influenced by two types of communication: the mass media and interpersonal channels. The mass media influence continually exists throughout the diffusion process but is more prevalent in the first half of the diffusion process of the innovation. The interpersonal communication channel expands in numbers during the first half but thereafter declines during the second half. The Bass Diffusion model (Bass, 1969) that represents rate of diffusion and the cumulative adoption of an innovation is shown in Equation 2 below:

Rate of diffusion

$$\frac{dN(t)}{dt} = (a + b\,N(t))\big(m - N(t)\big) \qquad (2)$$

Where,

'a' is the coefficient of external influence that represents the mass media communication channel, also called as the coefficient of innovation.

'b' is the coefficient of internal influence , imitation parameter which represents the interpersonal communication channel, also called as coefficient of imitation.

N (t) is the cumulative number of adopters at time t, and 'm' is the number of potential adopters of the innovation

3.2.3 Co-Diffusion Effects

When innovations diffuse, they do not just diffuse in vacuum, or exist in one, but they have positive or negative influence on the diffusion of an innovation (Peterson & Mahajan, 1978)(Peterson and Mahajan 1978). Peterson & Mahajan (1978) introduced a variant of the Bass diffusion model (Bass, 1969) incorporating more than a single innovation. The reason for developing this variant is that the diffusion of single innovation (new or existing) is not a self-

reliant phenomenon. Seldom is an innovation uninfluenced and a strong relationship may exist between two innovations over a long period of time

The rate of diffusion for the Bass diffusion model when co-diffusion effects are considered (Bucklin & Sengupta ,1993) is shown in Equation 3 and Equation 4:

$$\frac{dN_1(t)}{dt} = [a + bN_1(t) + cN_2(t)][\mathrm{m} - N_1(t)] \quad (3)$$

The cumulative penetration ratio is shown in Equation 4:

$$\frac{dF_1(t)}{dt} = [a + bF_1(t) + cF_2(t)][1 - \mathrm{F}(t)] \quad (4)$$

Where

N1 (t) is the cumulative adoption and F1 (t) is penetration ratio of innovation 1 at time t

N2 (t) is the cumulative adoption and F2 (t) is penetration ratio of innovation 2 at time t

F1 (t)=N1(t)/m

Parameter 'a' is the coefficient of external influence or the innovation parameter

Parameter 'b' is the coefficient of internal influence or the imitation parameter.

Parameter 'c' is the coefficient of co-diffusion which represents the impact of innovation2 on innovation1.

The diffusive interactions of two products can be represented using Equations 5 and Equation 6

$$innovation\ 1\ - F_1(t) = a1 + (b1 - a1 + 1) * F_1(t-1) - b1 * F_1^2(t-1) + c1 * F_2(t) - c1 * F_1(t-1) * F_2(t-1) \quad (5)$$

$$innovation\ 2\ - F_2(t) = a2 + (b2 - a2 + 1) * F_2(t-1) - b2 * F_2^2(t-1) + c2 * F_1(t) - c2 * F_2(t-1) * F_1(t-1) \quad (6)$$

Estimating the parameters in the equations above, assists in understanding the strength of the impact of each of the influence channels of the innovation adoption. The co-diffusion parameters ‘c’ and ‘c2’ captures two types of effects (Dewan, Ganley, & Kraemer, 2010). For instance, in Equation 5, when $c1 > 0$, it indicates a complementary effect of innovation2 on innovation1 and when $c1 < 0$, it indicates a substitutive effect of innovation 2 on innovation 1. The co-diffusion effect can be symmetric as well as asymmetric. When both c1 and c2 are positive, a complementary effect exists between the two innovations. The complementary effect indicates that increased adoption of one innovation enhances the adoption of the other over time and vice versa. A good example for this is the adoption of washers and dryers. When both c1 and c2 are negative, substitutive effects exists between the two innovations. The substitutive effect indicates that increased adoption of one innovation reduces the adoption of the other. A good example for this is the competition between different brands of automobiles.

3.3 Literature Review

3.3.1 Diffusion in IS

Diffusion research in information systems (IS) has accumulated an impressive body of studies over the years focusing on the diffusion of technologies, development methods, information systems etc. and identifying the drivers of diffusion of technologies. Diffusion studies analyze about how fast or slow ideas, products, innovations, and opinions take off and spread into the society (Valente, 1995). They facilitate the prediction of sales growth of new products and innovations (Peterson & Mahajan, 1978) through quantitative and qualitative methods to understand the evolution and adoption of innovations. Thus, innovation diffusion theory became a popular reference theory for empirical studies of information technologies. Diffusion theory is profoundly used by researchers in different fields to understand the spread of agricultural practices, technology, fertility control methods, policy innovations, consumer products, educational curricula, political reforms, and health promotion programs (Rogers, 2010).

Innovation diffusion studies facilitate understanding various aspects such as the previewed characteristics of innovations, the communications channels through which innovation take place, the type of social system, the adopter categories, stages of innovation diffusion etc. Premkumar, Ramamurthy, & Nilakanta (1994) leveraged on the innovation diffusion theory to understand the adoption of electronic data interchanges (EDI) for purchase orders, notices, billings, etc. by finding the influence channels of EDI diffusion which was a very new concept at the time. The authors found that the diffusion process of EDI was a combined effect of internal as well as external diffusion and stated that relative advantage as perceived by the individuals was a predictor of internal diffusion and technical capability was the main predictor of external influence. Beynon-Davies & Williams (2003) identify information systems development methods as a knowledge - based innovation systems. The authors identified that the Computerization Movement Organization(CMO)'s external influence and the recruitment of organizational members to such organizations as well as links to other CMO's through internal and external consultants both promote the diffusion of the information systems development methods (internal influence) such as rapid application development, agile development methods etc.

Zhu, Dong, Xu, & Kraemer (2006) studied about the determinants of the post-adoption usage of the innovation diffusion using innovation characteristics (relative advantage, compatibility, costs and security concern) and contextual factors (technology competence, organization size, competitive pressure and partner readiness). The authors noted that innovation diffusion can better be understood with a combination of innovation and contextual factors rather than by studying them separately. The authors also stated that economic and regulatory factors can result in uneven innovation diffusion. Garg, Smith, & Telang (2011) developed an empirical approach for measuring information diffusion and discovery in online social networks that have measurement challenges. The authors tested this scenario on online music community with data from 4000 online users and found that peers in such online communities significantly increase music discovery.

Thus, diffusion studies have been quite popular in the information systems domain exploring about various factors that facilitate or reduce the diffusion of technological innovations.

3.3.2 Diffusion Studies in Software Sourcing

Prior research on software sourcing has accumulated an impressive body of literature, given the wide adoption of the software sourcing models in the software development. IS Scholars have explored about various software sourcing options to unearth the factors that drive software sourcing adoption in organizations, to identify the factors that influence the choice of software sourcing, the benefits of adoption, risks, and impacts on the performance (Dibbern, Goles, Hirschheim, & Jayatilaka, 2004; Lacity, Khan, Yan, & Willcocks, 2010; Schneider & Sunyaev, 2016). Few other scholars investigate about the client-vendor relationship, project management, communication and coordination issues between the client and vendor, choosing the vendor locations in case of offshoring etc. (Bapna, Barua, Mani, & Mehra, 2010; Han, Lee, Chun, & Seo, 2013; Khan, Qurashi, & Khan, 2011). But IS scholars have constantly noted that software sourcing literature has reached substantial bounds with the huge number of snap shop studies and that there is a necessity to study about the spread of software sourcing phenomena over time (Alsudairi & Dwivedi, 2010; Aspray et al., 2009; Yadav & Gupta, 2008).

Scholars identified IS outsourcing as an innovative governance mechanism (Hu et al., 1997; Loh & Venkatraman, 1992) in the early 1990's and adopted the innovation diffusion theories to study the diffusion phenomenon of software sourcing arrangements. One of the oldest and notable studies existing in the software sourcing literature which applied the innovation diffusion theory is authored by Loh & Venkatraman (1992) during the time when IT outsourcing has emerged as a promising innovation in an organizations IT strategy. In this study, the authors used innovation diffusion theories to explore the influence sources for the outsourcing phenomenon by obtaining information about 60 outsourcing contracts during 1988 to 1990. The authors found that the outsourcing diffusion was the result of only internal influence channels. The study also considered the Kodak effect (a swift announcement by IBM that the work of Kodak is outsourced

after which IT outsourcing became a serious strategic choice for firms) as a critical event and explored about the strength of the internal influence channels post and pre-Kodak effect. The authors found that the internal influence was more pronounced post-Kodak regime and not pre-Kodak regime. Results from their study suggested that IT outsourcing mainly occurs through imitation where organizations mimic other organizations during uncertainties. But, a re-evaluation study of the influence sources of information systems outsourcing study that was conducted by Hu et al. (1997) revealed contrasting findings. The authors used a sample of 175 firms that outsourced there IS functions during the period from 1985 to 1995 to identify the best diffusion model that characterizes the diffusion of IS outsourcing. Results from their study contradicted the conclusions of Loh & Venkatraman (1992) that IS outsourcing diffusion is the result of mixed influence (both external and internal influences) and that the Kodak effect did not exist in the IS diffusion process. The authors empirically found evidence that external influence in the outsourcing decision is very prominent and the influence is exerted aggressively on managers by the vendors and the extensive information available in the trade press. The authors also found that the decision makers are also influenced by the communication among the organizations and their managers that may be considering or have adopted IS outsourcing.

In an analysis of the real life case studies of offshore outsourced projects, Chakrabarty (2006) noted that offshore outsourcing phenomenon can be considered as an organizational or administrative innovation process that is adopted by various clients and vendors. Chaudhury & Bharati (2008) leveraged on the innovation diffusion theory and institutional theory to conceptualize the factors that contribute to the adoption of IT services outsourcing by small and medium enterprises which is considered as a management innovation. The authors performed a literature survey and proposed that the adoption rate of IT sourcing by the vendors depends on three important factors 1) innovation profile features such as relative advantage and complexity of services 2) innovator profile of the vendor such as prestige level, firm size, education level and thirdly, 3) field level characteristic such as interfirm connections, service professionalism etc. The

authors proposed some valid propositions but a follow up empirical analysis was not performed by any other researchers to validate the propositions.

In another research by Mann, Kauffman, Han, & Nault (2011) the authors modeled the diffusion of outsourcing via announcements about IT outsourcing deals that exceeded 1 billion US dollars, happened during the time period 1999 to 2007 to test the existence of contagion effects in IT outsourcing. The authors performed their analysis in two stages. Firstly, the authors empirically tested whether IT outsourcing follows a pure diffusion process at the firm level as well as industry level, by estimating a log normal distribution. After they confirmed this, the authors tested to evaluate whether the contagion effects are present in the data. Contagion effects as considered by the authors are the effects such as large dollar mega deals acting as precipitating events for outsourcing adoption and large firms acting as examples for small firms by reducing the inhibitions about outsourcing adoption. The authors found that the mixed influence model best explained the IT outsourcing diffusion and the internal communication channels played an important role. They also found that the diffusion curves were asymmetric, left-skewed and flat. This suggests the existence of contagion effects where the firms across different industries were predetermined to adopt the IT outsourcing during the earlier stages.

Leveraging on the previous studies, it is evident that the software sourcing phenomena is proposed as organizational and administrative innovations in line with Rogers's definition of innovation which can be any practice, product, reform or any idea. Therefore, the study considers all the software sourcing phenomena: in-housing, outsourcing, offshore and on-shoring as organizational innovations.

In a recent study, Lewin & Volberda (2011) discussed the emergence of global sourcing of business services. The authors call for a need to understand the decision to offshore work and the overall growth of business service offshoring at a much nuanced level. The authors note that the literature on outsourcing has reached substantial proportions by analyzing the offshoring decisions based on single theme based explanations such as accumulation of experience, assessment of risks, costs, international exploitation of knowledge which provide only the

incremental explanations or factors emphasizing on the external factors for global sourcing. They state that there is no international business theory that can explain how and why firms offshore and develop over time as they do. Moreover, the authors note that the IS scholars have not addressed the interrelationships between firm level offshore decisions. They suggest the researchers to focus on co-evolutionary models of offshoring since offshoring dynamics is not an outcome of strategic decisions or environmental selection but a joint outcome of emergence, managerial intension, and environmental effects. In this study, the co-diffusion effects of software sourcing phenomena between traditional sourcing options such as in-housing and on-shoring to the later year innovative sourcing options such as offshoring and outsourcing are tested.

3.3.3 Co-diffusion Studies

While there was considerable prior research on diffusion of IT innovations (Rai, Ravichandran, & Samaddar, 1998; Teng et al., 2002) and few studies on the diffusion of software sourcing, most researchers were biased towards investigating innovations as if it were independent of other innovations (Colombo & Mosconi, 1995; Ladrón-de-Guevara & Putsis, 2015; Rogers, 2010). But, it is important to understand that innovations diffusing in a social system are interdependent. While diffusion studies explain the diffusion variables independently, in reality, they exert their effects on the process of diffusion interactively either potentiating or mitigating. The relative weight of each variable may change according to the circumstances characterizing the innovation and its context (Wejnert, 2002).

It is imperative to note that diffusion effect is not the sole reason for the dissemination of innovation as new products seldom are unaffected by the other existing or new products. It will be considered as a limitation if it is assumed that innovations diffuse on their own because products have diffusive interactions with the other existing or new products. The introduction of a innovation not only delivers the envisioned benefits but might affect the diffusion of another existing or new innovation due to interactions between them called as co-diffusion effects. Bucklin & Sengupta (1993) define co-diffusion as the interaction between the demands of innovations that

have separate adoption tracks. Sometimes co-diffusion effects between innovations are stronger than innovation effects.

Mahajan & Peterson (1985) extended the basic diffusion model proposed by Bass to incorporate the co-diffusion effects and tested the diffusion models to understand the type of co-diffusion effect between black and white, and color television. The authors found complementary and substitutive effects between the two innovations. The parameter estimate from the co-diffusion model suggested that growth in sales of color televisions had a substitutive effect on the sales of the growth of black and white television and the reverse effects were complementary.

Colombo & Mosconi (1995) suggested that single equation models that do not consider technological interactions are erroneous, suffering from omitted variables casting serious questions on studying individual innovations in isolation. Volberda & Lewin (2003) performed a co-evolutionary analysis between firms and suggest that adaptation and selection of innovations are not orthogonal forces but are fundamentally interrelated. They affirm that co-evolution explains the longitudinal time series adaptation within a historical context where changes in any one variable may be caused endogenously by changes in the other. Kim, Chang, & Shocker (2000) developed a dynamic model that captures both inter-product category effects and technological substitution within a broadened concept of a competitive IT market. They asserted empirically that sales of one related product category have an impact on another's market potential and, consequently, affects its market growth suggesting the existence of co-diffusion effects.

Bucklin & Sengupta (1993) highlighted the phenomenon of co-diffusion, and they posit that the diffusion of complementary innovations cannot be studied in isolation. The authors note that complementary innovations are those innovations when one or both can be acquired without the other (ex: Televisions – VCR; washers –dryer, computers - software's). Here the innovations are quasi- independent where the adopters of one innovation are not obliged to adopt the other. They note that the value of a focal innovation increases with the increased number of adoptions of a complementary innovation until saturation. They found that Universal Product Code symbols printed on packages of stocked items and laser scanners in supermarkets have asymmetric two-

way complementary co-diffusion effect and that the co-diffusion effects are stronger than the innovation effects. Their results provide support that the co-diffusion effect of UPC symbols on the bar scanners is stronger than the effect of scanners on the UPC symbols. Thus, indicating that the promotion of usage of UPC symbols automatically enhances the adoptions of scanners.

Dewan et al. (2010) examined the cross country diffusive interactions of personal computers and internet by applying the co-diffusion theories. The authors found that the co-diffusive effects are two-way complementary in nature and the impact of personal computers on the internet is stronger in developing countries compared to the developed ones. They identified that the installed computer base provides a boost to the rate of diffusion of the co-diffusive effect of the PC-installed base on internet diffusion is a factor that serves to narrow the divide internet penetration over time. Their findings suggest the policy makers that investments in the older generation of IT will bring greater diffusion benefits of newer technologies than the efforts to directly stimulate only the latest generations.

In a recent study, Niculescu & Whang (2012) explored the parallel market evolution of mobile wireless voice and wireless data services and examined the differences and complex interactions between the associated adoption processes in Japanese wireless market. Although wireless voice and data services were introduced with a gap individuals assimilated the usage of voice services faster than the data services. Moreover, the voice services remained the same over a long time, but data services evolved significantly as more content is ported to mobile platforms. After the introduction of data services, there was a huge competition between the participating player's wireless carriers, handset, manufacturers, and data service providers. The authors identified a need to coordinate across the participating players to improve their profit levels and also serve their customers. To understand the adoption process and the interactive interactions of the two services they used a discrete-time multiproduct model that accounts for diffusion effects and dynamic evolution of the services. The authors observed the existence of both direct network/imitation effects as well as two-way positive co-diffusion effects at the speed of adoption level. They found that the willingness of voice consumers to consider adopting data services is

positively related to both time and penetration of 3G handsets among voice services adopters. The results suggest the mobile content service providers to efficiently coordinate and strategize their offerings.

Jin-Xing, Xinping, & Siqing (2013) analyzed the co-diffusion patterns of the mobile IM service and the desktop IM service of China Mobile by applying the co-diffusion models proposed by Mahajan and Peterson using Non-Liner Seemingly Unrelated Regression (NL-SUR). Their results provided support that although Mobile IM is relatively new, the ease of use of the desktop IM lead to higher rate of diffusion of Desktop IM, with stronger innovation and imitation effects. Their co-diffusion analysis revealed that mobile IM service and the desktop IM service are both complementary and substitutive in nature Mobile IM has a substitutive effect on the desktop IM whereas desktop IM has a complementary effect on Mobile IM.

Another similar recent study by Westland, Hao, Xiao, & Shan (2016) analyze about the process of substitution as older technology is gradually replaced by latest technologies by extending the Bass mathematical articulation of Rogers's diffusion model using data from 2007 to 2011. They analyzed the substitutive, complementary and the network effects of three complementary instant messaging services provided by China mobile between Short Message Services (SMS), Multimedia Instant Messaging (MIM), and Instant Messaging on personal computers (PC-IM). They found that the adoption of SMS and MIM or simultaneous adoption of the two services negatively impacted the PC-IM adoption to reduce. The authors found the support that the innovation effect is negligible and that its influence is overwhelmed by the network effects from its substitutes and complement services. The authors note that contagion effects provide an accurate description of the mechanisms under the network effects in the form of network effects of other substitutive or complement services.

3.3.4 Research Gap

Thus, the previous diffusion and co-diffusion studies provide valuable insights for the developers, users and the providers of the innovations about the dynamics of the adoption patterns,

the influence channels, and the contagion effects. The existing literature on diffusion has largely focused on diffusion as an independent process ignoring the complementarities of diffusion (Ladrón-de-Guevara & Putsis, 2015). It is also essential to note that the co-diffusion effects are important since connectivity among innovations improves or reduces the end-user value of the products. Organizations can benefit from the co-diffusion effects as it helps to gain insights about innovation to promote and identify the potential partners to obtain a mutual benefit through co-marketing and also assists in making decisions about choosing the innovation that has great future. It is also observed that the prevalence of the co-diffusion studies has risen over the past five years and it can be expected that the studies in this area will continue to increase in the future. The insights gained from the co-diffusion analysis along with the diffusion studies help the stakeholders to make informed decisions about strategizing the offerings coordination, current, and future trends of the innovations by empirically validating the objective data etc. However, review of the existing diffusion, co-diffusion, and software sourcing literature revealed that the diffusion and co-diffusion studies are lacking to understand the trends of adoption and diffusion of the software sourcing phenomena over time. Moreover, the recent changes in the trends of outsourcing and offshoring in the form of nearshoring and back shoring call for the need to reevaluate the adoption trends of the existing and prevalent sourcing options such as outsourcing and offshoring. Though this study, the gap is filled by answering the following research questions.

Research Questions

Research Question 1: Do co-diffusion effects exist between 1) on-shoring and offshoring 2) in-housing and outsourcing.

Research Question 2: What is the nature of the co-diffusion effects between 1) offshoring – on-shoring and 2) in-housing -- outsourcing?

3.4 Hypotheses

3.4.1 In-House and Outsourced Software Development Projects

According to Lankford & Parsa (1999) , "Outsourcing is defined as the procurement of products or services from sources that are external to the organization" whereas in-housing is performing activities internal to the organization. Outsourcing reduces the costs and increases assets by moving activities outside the firm (Abramovsky & Griffith, 2006). The decision about choosing outsourcing versus in-housing depends on the relative cost of producing the services in-house compared to outsourcing the same activity (Abramovsky & Griffith, 2006). Some organizations choose to retain some capability and capacity in-house and outsource part of the activity (Harland, Knight, Lamming, & Walker, 2005). One of the reasons for outsourcing is to improve management focus and access technical talent not available in-house (Lacity & Willcocks, 1998; Levina & Ross, 2003). Given the benefits of outsourcing over in-housing, it is expected that organizations that perform core activities in-house will identify the benefits of outsourcing the non-core activities resulting in a positive enhancing effect for the outsourcing. Whereas outsourcing higher number of software projects may have a negative impact on the adoption of in-housing, firms may slowly start getting used to the benefits of outsourcing and may find it difficult to in-house. Therefore, substitutive effect is postulated for the co-diffusion effect of outsourcing on in-house activities.

Given this, the Hypothesis 1 and 2 are proposed:

H1: The co-diffusion effect of in-house on outsourcing is complementary.

H2: The co-diffusion effect of outsourcing on in-housing is substitutive.

3.4.2 Onshore and Offshore Software Development Projects

Offshore sourcing is the economization of more expensive on-shore resources being replaced with cheaper offshore resources to reduce their baseline costs (Chua & Pan, 2008), while on-shoring is performing the activities with resources within the country. IT managers are constantly under pressure of delivering projects on time along with the responsibility of

maintaining projects costs within limit for which they have to find expert staff in fast-moving technologies (Carmel & Agarwal, 2006). However, it is unfortunate that on-shore staff typically will not accept a lower salary or accept for being relocated offshore (Chua & Pan, 2008) due to which the managers choose offshore sourcing strategy for the benefit of the projects. Hence, it is estimated that higher adoption of on-shoring will propel the managers to think about other sourcing options that can reduce project costs such as offshoring. Once the companies start offshoring their projects and reap benefits in the form of offshoring, more companies would want to join them hence higher adoption of offshoring would reduce the adoption of on-shoring. Hence, it is postulated that higher adoption of offshoring will expose the information technology managers to more difficulties than benefits making them rethink about the offshore decision causing a complementary effect on on-shoring in the form of back-sourcing. Hence, the following hypotheses are proposed

H3: The co-diffusion effect of on-shoring and offshoring is complementary.

H4: The co-diffusion effect of offshoring on on-shoring is substitutive.

TABLE 3.1: SUMMARY OF HYPOTHESES

H.No	Hypotheses of the study
H1	The co-diffusion effect of in-house on outsourcing is complementary.
H2	The co-diffusion effect of outsourcing on in-housing is substitutive.
H3	The co-diffusion effect of on-shoring and offshoring is complementary.
H4	The co-diffusion effect of offshoring on on-shoring is substitutive.

3.5 Data and Research Method

Data for the study is obtained from the ISBSG-Release 12 and Release 13 software project repository containing information of the software projects developed in 20 different countries. The data in the repository has been reported voluntarily by the various software organizations and industry leaders. It has also been validated based on ISBSG quality guidelines and was used in

some of the previous studies (Bagchi, Kirs, Udo, & Cerveny, 2015; Jiang & Naudé, 2007; Oligny, Bourque, Abran, & Fournier, 2000). The data used to analyze the diffusion process is the year of adoption of the software sourcing obtained from the data repository. The year of adoption of software sourcing is obtained from the year of implementation of the software development projects that were developed with a specific type of sourcing phenomena. The year of implementation as reported by the ISBSG data repository is the year in which the project was implemented if known or other dates such as project end date, project start date, estimated implementation date. Table 3.2 shows the number of software projects developed with a specific type of software sourcing arrangement. For each of the four types of software sourcing, the information about the year of implementation of the projects is aggregated to find the number of projects that adopted a specific type of software sourcing over the years. Using this the cumulative adoption of each of the software sourcing phenomena is obtained over the years.

TABLE 3.2: NUMBER OF PROJECTS

Software Sourcing	Number of projects
Onshore	2814
Offshored	167
In house	1269
Outsourced	1297

The cumulative adoption of software sourcing alignments over the years is displayed in Figure 3.2. This information is used to perform co-diffusion analysis using the software package - SHAZAM that is used to test econometric and statistical models. Since the systems of equations that represent diffusion and co-diffusion process are non-linear equations with have auto correlated errors, Non Linear - Seemingly unrelated regression (NL-SUR) is used.

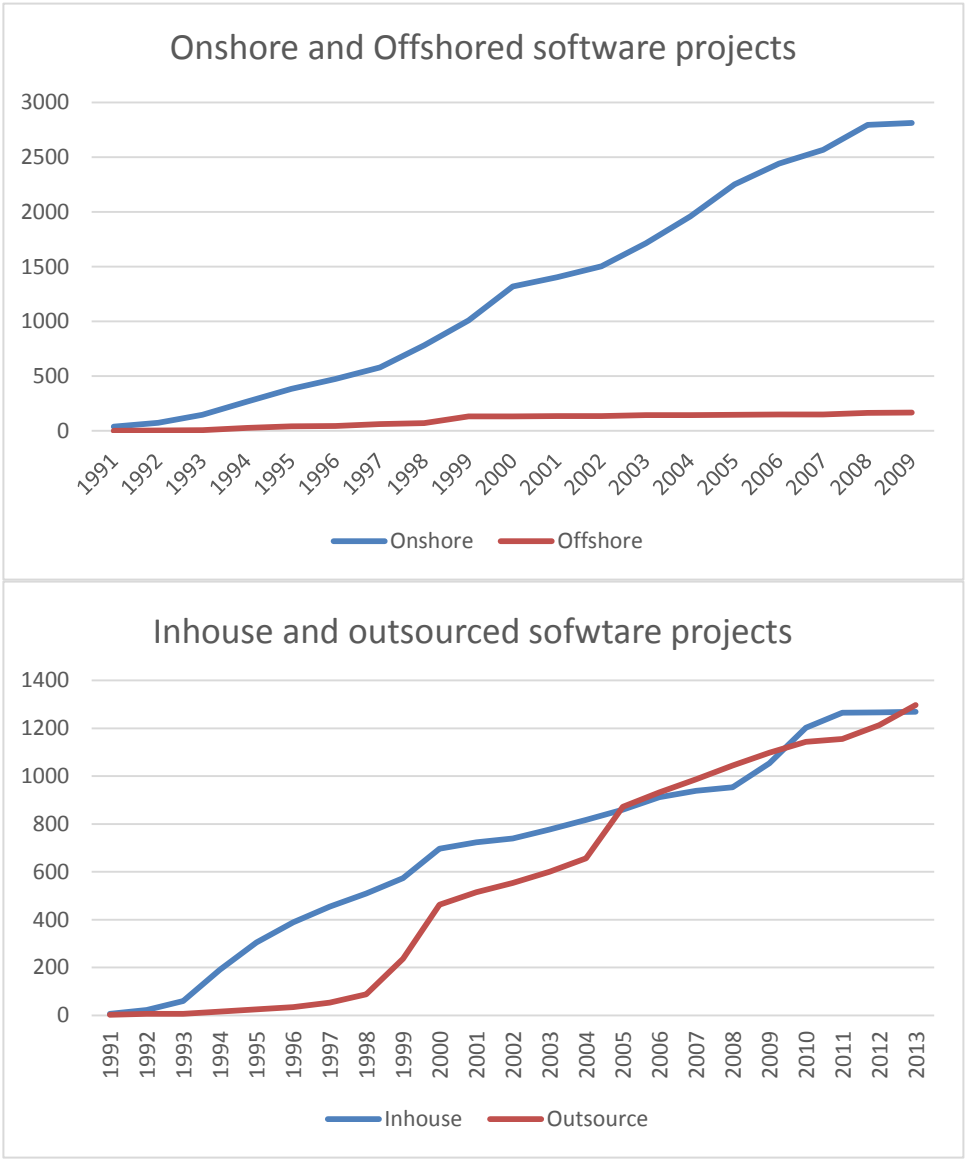


Figure 3.2: Cumulative Adoption over the Years

Non-linear seemingly unrelated regression (SUR) is used to estimate (a) the coefficient of innovation, (b) coefficient of imitation and (c) the coefficient of co-diffusion consistent with the literature (Cho, 2015; Dewan et al., 2010; Jin-Xing et al., 2013). Nonlinear Seemingly unrelated regression with SHAZAM estimates the coefficients for a set of nonlinear equations using a variable metric method (Diana, Kenneth J., David, & Madeleine, 2011).

3.6 Results

The parameter estimates of the co-diffusion analysis of the study are shown in Table 3.3. It is observed that one-way co-diffusion effects exist between in house and outsourced software projects (c2= 0.2307 significant at $p<0.05$). The co-diffusion effect of in-house development on the outsourced development is found to be complementary and significant and the vice versa is found to be substitutive yet not significant. Thus, supporting the H1 and not supporting the H2. In addition to the co-diffusion effects, it is found that outsourcing diffusion also occurs through imitation effects (b2 =0.1612 significant at $p<0.05$) via peer to peer communication between managers and organizations that may have been considering the adoption or have already adopted the outsourcing. But in-housing adoption spreads only through external influence channels /innovation effects (a1 = 0.0061 significant at $p<0.05$) via information obtained from mass media communication or the trade press about the in-house projects that have been successful or unsuccessful.

TABLE 3.3: DIFFUSION AND CO-DIFFUSION COEFFICIENTS

Software sourcing	Coefficient	Value	Software sourcing	Coefficient	Value
In-house	a1	0.0061**	**Outsourced**	a2	-0.0005
	b1	-0.0187		b2	0.1612 **
	c1	-0.0279		c2	0.2307**
On-shore	a1	0.0076**	**Offshore**	a2	0.0013**
	b1	-0.0706		b2	-0.0669
	c1	1.9797**		c2	0.0012

***significant at 0.1 level; ** significant at 0.05 level; *significant at 0.001 level

It is also observed that one-way co-diffusion effects exist between on-shore and offshore software development. The co-diffusion effect of on-shore development on the offshore development is found to be complementary but not significant, not supporting H3. But the co-diffusion effect of offshoring on on-shoring is found to be complementary and significant (c1 = 1.9797 significant at $p<0.05$) . The co-diffusion effect of offshoring on on-shore software

development is in contrast to the substitutive relationship as expected. Thus, not supporting the H4. In addition to the co-diffusion effects of offshoring on the on-shoring diffusion, on shoring also diffuses due to its innovation effects/ external influence (a1=0.0076 significant at $p<0.05$) from the mass media communication effects and from the trade press about the on-shoring adoption and its success. It is found that offshoring diffusion occurs only through innovation effects (a2 =0.0013 significant at $p<0.05$) via the external influences. The results are discussed in detail in the following section. Results of the hypotheses is shown in table 3.4.

TABLE 3.4: RESULTS OF HYPOTHESES

H.No	Hypotheses of the study	Result
H1	The co-diffusion effect of in-house on outsourcing is complementary	Supported
H2	The co-diffusion effect of outsourcing on in-house is substitutive.	Not supported
H3	The co-diffusion effect of on-shoring on offshoring is complementary.	Not Supported
H4	The co-diffusion effect of offshoring on on-shoring is substitutive.	Significant but not Supported

3.7 Discussion

Results from the analysis suggest the existence of co-diffusion effects between the in-house - outsource software sourcing diffusion and between on-shore - offshore software sourcing and the effects are only one way co-diffusion effects. Further analysis of the results of the in-house - outsourcing suggest that, in-house diffusion occurs only due to the external influences whereas outsourcing diffusion is the result of both the internal effects and the co-diffusion effects. In-house diffusion is a traditional software development method which is generally chosen if the organization has the capacity to take up the additional IT development responsibility. The spread of in-housing adoption over the years occurs through the influences that the managers have due to the information obtained through mass media and trade press about in housing success or failures through the trade press or mass media communication channels. The outsourcing diffusion occurs

mainly through co-diffusion effects through interpersonal communication between the managers who have adopted or planning to adopt outsourcing. Results also suggest that higher adoption of in-housing enhances the adoption of outsourcing. This is possible because higher in-housing indicates that more number of organizations looking forward to implement new technologies or software's to automate their process. Thus, an influx in the in-house development indicates a higher dependency on the IT assets by the organizations and high dependency on the IT assets automatically increases the number of software development projects which would require dedicated experts who can work on the projects quickly with increased efficiency and less risk which can be achieved through outsourcing. Thus, higher adoption of in-housing increases the adoption of outsourcing. Outsourcing has other added benefits where the non IT organizations can focus on their core competencies by outsourcing their IT work. Given the added benefits of outsourcing there will be an increase in the outsourcing adoption. The reverse relationship of the effect of outsourcing on in-house projects was found to be non-significant suggesting that there is no negative effect of outsourcing on offshoring. Therefore, it can be expected that in-housing and outsourcing are both on a rise where one type of sourcing is enhancing the other. Thus in the near future it can be expected that there will be increasing trends in the adoption of both in-housing and outsourcing as well.

Analysis of the results of the onshore-offshore sourcing suggest that, both the on-shore diffusion and offshore diffusion occur due to the external influences though information obtained in the trade press, mass media communication channels etc., but not through interpersonal communication between the prior and potential adopters. Results also suggest that higher adoption of onshore does not impact the offshoring adoption, whereas higher adoption of offshoring enhances the adoption of on-shoring. The co-diffusion effect of offshoring on on-shoring is in contrast to what was expected. Although offshore sourcing has become increasingly popular and important in reducing the costs of software development, and provides access to world class expertise to develop the software's, it is creating uncertainties in the IT workforce leading to the failures of the offshored projects (Erber & Sayed-Ahmed, 2005) . Offshore development adds new

facets to development such as culture, distance, time, and space challenges in communication channels and infrastructure that can complicate development can lead to project failures (Christiansen, 2007). The benefits of offshoring is low-cost labor but the damages that offshoring can cause are missed deadlines, dissatisfied users, and failure to reduce development costs. In the recent times it is found that many organizations have faced offshoring failures due to various issues such as cultural clashes, communication issues, inability build necessary social and human capital, challenges with domain knowledge, lack of commitment of external developers (Moe et al., 2014). In a literature study performed by Smite & Wohlin (2011), it is found that offshoring experiences of organizations resulted in failures due to the challenges faced by the companies, thus indicating that offshoring software development can be very challenging. Carmel & Tjia (2005) and Ebert (2007) also note that companies have realized that the cost savings are small and problems are bigger compared to the co-located development. Thus, the study results strengthen the argument provided by the previous scholars that offshoring is slowly reducing thus enhancing the adoption of onshore software development. The managerial implications the current study are discussed in the following section.

3.8 Managerial Implications

The study has an extensive application to the information technology and software development industry since many organizations are involving in global market interactions to obtain software technology and for the software development activities. The study has important implication to the managers of the organizations who are involved in the decision making about sourcing the IT assets of their organizations. Complementarities of diffusion of software sourcing gives insights about future trends of the diffusion process of the software sourcing phenomena. With respect to the sourcing options, the top management of organizations could understand the future of the different sourcing options and help make a better decision while choosing a suitable sourcing option for their projects.

3.9 Future Research

Co-diffusion studies assist in understanding the current and future trends in the adoption practices of technologies and practices. The current research focusses on co-diffusion analysis of sourcing options for software development projects developed in multiple countries. The current research will be expanded by testing how the co-diffusion trends vary with large sized and small sized projects. Also this co-diffusion effects can be tested for the diffusion process within and outside a country to understand the diffusion process to a specific country. The study could also be expanded to test the co-diffusion effects of other development aspects like software development methodologies such as waterfall development, agile development and other technological aspects used in the software development process.

3.10 Conclusion

The study considers software sourcing arrangements such as offshoring and outsourcing as strategic organizational innovation and analyzes the temporal diffusion of software sourcing arrangements by applying the innovation diffusion theories. Specifically, the study tests the existence of the co-diffusion effects of related yet independent innovations that have separate adoption tracks: 1) between on-shoring and offshoring and 2) between in-housing and outsourcing. The results from the analysis indicate the existence of one way co-diffusion effects and also indicate that the co-diffusion effects are stronger than the diffusion effects. The co-diffusion effect of in-house development on the outsourced development is found to be complementary and significant and the vice versa effect was not found to be significant. Thus indicating that in-house diffusion occurs only due to the external influences whereas outsourcing diffusion is the result of both the internal effects and the co-diffusion effects. The co-diffusion effect indicates that the higher adoption of in-housing enhances the adoption of outsourcing. Also, the co-diffusion effect of on-shore development on the offshore development does not exist, but the co-diffusion effect of offshoring on-shoring is found to be complementary. This indicates that the higher adoption of onshore does not impact the offshoring adoption, whereas higher adoption of offshoring enhances

the adoption of on-shoring. Finally it can be concluded that outsourcing and on-shoring are going to have increased adoption by organizations and would be considered as prospective sourcing options given the positive boost from the in-house and offshore sourcing. This analysis assists in understanding the current and future trends in the adoption practices of technologies and practices.

3.11 References

Abramovsky, L., & Griffith, R. (2006). Outsourcing and offshoring of business services: How important is ICT? *Journal of the European Economic Association*, 4(2-3), 594-601.

Ågerfalk, P. J., Fitzgerald, B., & Stol, K. (2015). Not so shore anymore: The new imperatives when sourcing in the age of open. Paper presented at the *Ecis,*

Alsudairi, M., & Dwivedi, Y. K. (2010). A multi-disciplinary profile of IS/IT outsourcing research. *Journal of Enterprise Information Management, 23*(2), 215-258.

Aspray, W., Mayadas, F., & Vardi, M. Y. (2009). 3. Globalization and off shoring of software. *Innovation Imperative: National Innovation Strategies in the Global Economy,* 24.

Bagchi, K., Kirs, P., Udo, G., & Cerveny, R. (2015). Characteristics and determinants of insourced and offshored projects: A comparative analysis. *Journal of World Business, 50*(1), 108-121.

Bapna, R., Barua, A., Mani, D., & Mehra, A. (2010). Research commentary—Cooperation, coordination, and governance in multisourcing: An agenda for analytical and empirical research. *Information Systems Research, 21*(4), 785-795.

Bass, F. M. (1969). A new product growth for model consumer durables. *Management Science, 15*(5), 215-227.

Beynon-Davies, P., & Williams, M. D. (2003). The diffusion of information systems development methods. *The Journal of Strategic Information Systems, 12*(1), 29-46.

Bucklin, L. P., & Sengupta, S. (1993). The co-diffusion of complementary innovations: Supermarket scanners and UPC symbols. *Journal of Product Innovation Management, 10*(2), 148-160.

Carmel, E., & Agarwal, R. (2006). The maturation of offshore sourcing of information technology work. *Information systems outsourcing* (pp. 631-650) Springer.

Carmel, E., & Tjia, P. (2005). *Offshoring information technology: Sourcing and outsourcing to a global workforce* Cambridge University Press.

Chakrabarty, S. (2006). Real-life case studies of offshore outsourced IS projects: Analysis of issues and socio-economic paradigms.

Chaudhury, A., & Bharati, P. (2008). IT outsourcing adoption by small and medium enterprises: A diffusion of innovation approach.

Cho, D. (2015). An empirical analysis of smartphone diffusions in a global context.

Christiansen, H. M. (2007). Meeting the challenge of communication in offshore software development. Paper presented at the *International Conference on Software Engineering Approaches for Offshore and Outsourced Development,* 19-26.

Chua, A. L., & Pan, S. L. (2008). Knowledge transfer and organizational learning in IS offshore sourcing. *Omega, 36*(2), 267-281.

Colombo, M. G., & Mosconi, R. (1995). Complementarity and cumulative learning effects in the early diffusion of multiple technologies. *The Journal of Industrial Economics,* , 13-48.

Contractor, F. J., Kumar, V., Kundu, S. K., & Pedersen, T. (2010). Reconceptualizing the firm in a world of outsourcing and offshoring: The organizational and geographical relocation of high- value company functions. *Journal of Management Studies, 47*(8), 1417-1433.

Dewan, S., Ganley, D., & Kraemer, K. L. (2010). Complementarities in the diffusion of personal computers and the internet: Implications for the global digital divide. *Information Systems Research, 21*(4), 925-940.

Diana, W., Kenneth J., W., David, B. & Madeleine, G. (2011). SHAZAM reference manual. Retrieved from http://store.econometrics.com/shazam/shazam_reference_manual_11_interior.pdf

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

Ebert, C. (2007). Optimizing supplier management in global software engineering. Paper presented at the *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference On,* 177-185.

Erber, G., & Sayed-Ahmed, A. (2005). Offshore outsourcing. *Intereconomics, 40*(2), 100-112.

Garg, R., Smith, M. D., & Telang, R. (2011). Measuring information diffusion in an online community. *Journal of Management Information Systems, 28*(2), 11-38.

Han, H., Lee, J., Chun, J. U., & Seo, Y. (2013). Complementarity between client and vendor IT capabilities: An empirical investigation in IT outsourcing projects. *Decision Support Systems, 55*(3), 777-791.

Harland, C., Knight, L., Lamming, R., & Walker, H. (2005). Outsourcing: Assessing the risks and benefits for organizations, sectors and nations. *International Journal of Operations & Production Management, 25*(9), 831-850.

Hu, Q., Saunders, C., & Gebelt, M. (1997). Research report: Diffusion of information systems outsourcing: A reevaluation of influence sources. *Information Systems Research, 8*(3), 288-301.

Jiang, Z., & Naudé, P. (2007). An examination of the factors influencing software development effort. *International Journal of Computer Information and Systems Science and Engineering, 1*(3), 182-191.

Jin-Xing, H., Xinping, X., & Siqing, S. (2013). Mobile operator's dilemma: An exploratory study on the co-diffusion of mobile IM and desktop IM of china mobile. Paper presented at the *E-Business Engineering (ICEBE), 2013 IEEE 10th International Conference On,* 250-256.

Kedia, B. L., & Mukherjee, D. (2009). Understanding offshoring: A research framework based on disintegration, location and externalization advantages. *Journal of World Business, 44*(3), 250-261.

Khan, A. I., Qurashi, R. J., & Khan, U. A. (2011). A comprehensive study of commonly practiced heavy and light weight software methodologies. *arXiv Preprint arXiv:1111.3001,*

Kim, N., Chang, D. R., & Shocker, A. D. (2000). Modeling intercategory and generational dynamics for a growing information technology industry. *Management Science, 46*(4), 496-512.

Lacity, M. C., Khan, S., Yan, A., & Willcocks, L. P. (2010). A review of the IT outsourcing empirical literature and future research directions. *Journal of Information Technology, 25*(4), 395-433.

Lacity, M. C., & Willcocks, L. P. (1998). An empirical investigation of information technology sourcing practices: Lessons from experience. *MIS Quarterly,*363-408.

Ladrón-de-Guevara, A., & Putsis, W. P. (2015). Multi-market, multi-product new product diffusion: Decomposing local, foreign, and indirect (cross-product) effects. *Customer Needs and Solutions, 2*(1), 57-70.

Lankford, W. M., & Parsa, F. (1999). Outsourcing: A primer. *Management Decision, 37*(4), 310-316.

Levina, N., & Ross, J. W. (2003). From the vendor's perspective: Exploring the value proposition in information technology outsourcing. *MIS Quarterly,* 331-364.

Lewin, A. Y., & Volberda, H. W. (2011). Co-evolution of global sourcing: The need to understand the underlying mechanisms of firm-decisions to offshore. *International Business Review, 20*(3), 241-251.

Loh, L., & Venkatraman, N. (1992). Diffusion of information technology outsourcing: Influence sources and the kodak effect. *Information Systems Research, 3*(4), 334-358.

Mahajan, V., Muller, E., & Bass, F. M. (1991). New product diffusion models in marketing: A review and directions for research. *Diffusion of technologies and social behavior* (pp. 125-177) Springer.

Mahajan, V., & Peterson, R. (1985). Models for innovation diffusion. *Sage Publication,*

Mann, A., Kauffman, R. J., Han, K., & Nault, B. R. (2011a). Are there contagion effects in information technology and business process outsourcing? *Decision Support Systems, 51*(4), 864-874.

Moe, N. B., Šmite, D., Hanssen, G. K., & Barney, H. (2014). From offshore outsourcing to insourcing and partnerships: Four failed outsourcing attempts. *Empirical Software Engineering, 19*(5), 1225-1258.

Niculescu, M. F., & Whang, S. (2012). Research Note—Co-Diffusion of wireless voice and data services: An empirical analysis of the Japanese mobile telecommunications market. *Information Systems Research, 23*(1), 260-279.

Oligny, S., Bourque, P., Abran, A., & Fournier, B. (2000). Exploring the relation betwe

en effort and duration in software engineering projects. Paper presented at the *Proceedings of the World Computer Congress,* 175-178.

Palugod, N., & Palugod, P. A. (2011). Global trends in offshoring and outsourcing. *International Journal of Business and Social Science, 2*(16)

Peterson, R. A., & Mahajan, V. (1978). Multi-product growth models. *Research in Marketing, 1*(20), 1-23.

Premkumar, G., Ramamurthy, K., & Nilakanta, S. (1994). Implementation of electronic data interchange: An innovation diffusion perspective. *Journal of Management Information Systems, 11*(2), 157-186.

Rai, A., Ravichandran, T., & Samaddar, S. (1998). How to anticipate the internet's global diffusion. *Communications of the ACM, 41*(10), 97-106.

Rogers, E. M. (2010). *Diffusion of innovations* Simon and Schuster.

Schneider, S., & Sunyaev, A. (2016). Determinant factors of cloud-sourcing decisions: Reflecting on the IT outsourcing literature in the era of cloud computing. *Journal of Information Technology, 31*(1), 1-31.

Scott, J. E. (2007). Mobility, business process management, software sourcing, and maturity model trends: Propositions for the IS organization of the future. *Information Systems Management, 24*(2), 139-145.

Smite, D., & Wohlin, C. (2011). A whisper of evidence in global software engineering. *IEEE Software, 28*(4), 15.

Teng, J. T., Grover, V., & Guttler, W. (2002). Information technology innovations: General diffusion patterns and its relationships to innovation characteristics. *IEEE Transactions on Engineering Management, 49*(1), 13-27.

Valente, T. W. (1995). Network models of the diffusion of innovations.

Volberda, H. W., & Lewin, A. Y. (2003). Co-evolutionary dynamics within and between firms: From evolution to co-evolution. *Journal of Management Studies, 40*(8), 2111-2136.

Wejnert, B. (2002). Integrating models of diffusion of innovations: A conceptual framework. *Annual Review of Sociology, 28*(1), 297-326.

Westland, J. C., Hao, J. X., Xiao, X., & Shan, S. (2016). Substitutes, complements and network effects in instant messaging services. *Networks and Spatial Economics, 16*(2), 525-543.

Yadav, V., & Gupta, R. K. (2008). A paradigmatic and methodological review of research in outsourcing.

Zhu, K., Dong, S., Xu, S. X., & Kraemer, K. L. (2006). Innovation diffusion in global contexts: Determinants of post-adoption digital transformation of European companies. *European Journal of Information Systems, 15*(6), 601-616.

Vita

Niharika Dayyala earned her Bachelor of Engineering degree in Civil Engineering from Andhra University, Visakhapatnam, India in 2006. In 2008, she received Master of Science in Information Technology from Jawaharlal Nehru Technological University, Hyderabad, India. In 2012, she joined the doctoral program in Business Administration with specialization in Information Systems at The University of Texas at El Paso.

Niharika Dayyala's research interests include diffusion studies, business value of IT, software development project performance, strategies for successful software development, software sourcing etc. She presented her research at conferences such as Decision Sciences Institute (DSI), Western Decision Sciences Institute Annual Meeting (WDSI) and Global Information Technology Management Association (GITMA). She received "Best Application Paper Award" for here research titled "Diffusion of IFRS using Innovation diffusion models" at the WDSI conference. She has also published a book chapter in the book titled "Entrepreneurship in Technology for the ASEAN".

Niharika Dayyala worked as an assistant professor at The University of Texas at El Paso for five years. She taught courses to the undergraduate students at the College of Business Administration. Her teaching interests include data analytics, database management, information systems security, business application programming and business statistics. Prior to coming to UTEP, Niharika Dayyala has worked for three years as a software development senior analyst at Dell International Services Pvt. Ltd. at Bangalore India. She worked on several data warehouse projects on building the Dell Data Warehouse. She has accepted a position as an assistant professor of Business Information Systems at the Illinois State University, Normal.

Permanent address:57-5-4/1, Mallikarjuna Nagar, Rajahmundry, India

This thesis/dissertation was typed by Niharika Dayyala.