Shared Experience, Outcome, and Forgetting: An Empirical Study

Keumseok Kang
*Florida International University*, kskang@fiu.edu

Inkyoung Hur
*Nova Southeastern University*, ihur@nova.edu

Follow this and additional works at: [http://aisel.aisnet.org/icis2017](http://aisel.aisnet.org/icis2017)
Shared Experience, Outcome, and Forgetting: An Empirical Study

Completed Research Paper

Keumseok Kang
College of Business
Florida International University
Miami, FL 33199
kskang@fiu.edu

Inkyoung Hur
College of Engineering and Computing
Nova Southeastern University
Fort Lauderdale, Florida 33314
ihur@nova.edu

Abstract

Shared experience (i.e., experience working together in prior projects) among project team members is known to affect the project team's performance. However, little is known of whether this effect is moderated by other factors. In this study, we examine the two attributes of shared experience as moderators: the outcome of shared experience and the time lapse of shared experience. We find that the effect of shared experience on project performance shows an inverse U-shaped curve with respect to the degree of failure (or success) of shared experience, suggesting that the effect of shared experience on project performance becomes strongest under intermediate degrees of failure (or success) of shared experience but comparatively weaker when the degree of failure (or success) is high or low. Moreover, we find that the inverse U-shaped curve becomes more flattened as the lapse of shared experience increases. This study makes a theoretical contribution by finding nonlinear moderating effects of shared experience. It also provides practical implications to software project management.

Keywords: Shared experience, outcome, forgetting, software development project

Introduction

Software development is an interdependent group task where a group of individuals with various specialties works together. Due to the interdependent nature of this group task, interrelationship and teamwork among project team members have been considered as one of the most influencing factors to determine the success of a software project (Alaranta and Betz 2012; Faraj and Sproull 2000; Qu et al. 2012; Staats 2012).

Factors related to teamwork and interrelationship have been extensively studied in academia. One of the factors is shared experience among project team members, which is sometimes referred to as experience working together (Reagans et al. 2005) or team familiarity (Huckman et al. 2009; Staats 2012). Shared experience shows positive effects on innovative tasks (Edmondson 1996; Gruenfeld et al. 1996; Lee et al. 2004; Sosa 2011) and organizational learning (Lewis et al. 2005). Recent studies on software development observe that shared experience among project team members helps project team members build and sustain a good interrelationship and teamwork and improve project performance (Faraj and Sproull 2000; Huckman et al. 2009; Reagans et al. 2005; Staats 2012). Researchers explain the effect of shared experience with the concepts of transactive memory (Lewis et al. 2005; Wegner 1987), coordinating ability (Faraj and Sproull 2000), and trust established among team members (Reagans et al. 2005).

Although prior studies on shared experience provide a significant contribution to both academia and practice, they overlook the outcome of shared experience, i.e., whether the projects in which team members have worked together before were successful or not. One natural question to ask is whether or not the effect of shared experience in a successful project and that in an unsuccessful (or failure) project are the same. By
working together, project team members seem to build effective transactive memory (i.e., knowledge about who knows what within the team) (Lewis et al. 2005; Wegner 1987) which creates a positive effect of shared experience on team performance. Transitive memory is developed irrespective of whether prior experiences are successful or not. However, when it comes to the trust among team members (Reagans et al. 2005), failures may deteriorate the trust among team members and create a bad impact on teamwork and performance. Hence, we believe that the outcome of prior shared experience differentially moderates the effect of shared experience on focal project’s performance by influencing team members’ behaviors differently.

It is known that knowledge cumulated from prior experience does not persist forever but depreciates over time in general. This phenomenon is referred to as knowledge depreciation (Argote 2012). In the literature, this knowledge depreciation has been originally examined at the individual level (Ebbinghaus 1964), but also observed at the organizational level as well (Argote 2012; Boone et al. 2008). Shared experience is not different. It is found that the effect of shared experience depreciates over time (Kang and Hur 2013). However, it is also known that knowledge depreciation does not always occur (Boone et al. 2008). People sometimes try to forget or not to forget some experiences than others. For example, people generally try to forget prior disastrous or shameful experiences (James et al. 2016), while they try to remember (not to forget) prior meaningful or important experiences (Boone et al. 2008). This intentional remembering or forgetting may facilitate or hinder knowledge depreciation caused by time lapse. We believe that the outcome of shared experience influences team members’ behaviors on intentional remembering and forgetting, and differentially affect the depreciation effect of shared experience over time.

In summary, in this study, first, we show how the effect of shared experience on the focal software project’s performance is moderated by the outcome of shared experience. Second, we show the effect of shared experience is also differentially depreciated over time by the outcome of shared experience. This study makes a contribution to the literature on shared experience by showing the moderating effects of the outcome of shared experience. Given that software development still suffers from a large number of project failures (El Emam and Koru 2008) and that contemporary software development practices require team work and coordination which benefits from shared experience among project team members (Faraj and Sproull 2000), understanding the nuanced effect of shared experience moderated by the outcome of shared experience is of practical significance, too.

**Theory and Hypotheses**

We use shared mental models as our theoretical foundation for explaining the moderating roles of the outcome of shared experience on the effect of shared experience on project team performance and its depreciation caused by time lapse. In this section, we explain what shared mental models are and how they are different from other possible theoretical concepts such as transactive memory, two main processes to build shared mental models (namely, knowledge transfer and integration), and the role of communications in these two processes. Then, we explain how the different outcomes of shared experience lead to different types and depths of communications, which in turn influence knowledge transfer and integration processes, which in turn affects building shared mental models and eventually team performance (Hypothesis 1). We further explain that the different outcomes of shared experience lead to different patterns of intentional forgetting which substitute for knowledge depreciation caused by time lapse (Hypothesis 2).

**Shared mental models**

Transactive memory (Lewis et al. 2005; Wegner 1987) and expertise coordination ability (Faraj and Sproull 2000) play a significant role in explaining the effect of shared experience. Huckman et al. (2009) and Staats (2012) show that shared experience among software project team members increases the project team’s performance by building effective transactive memory (i.e., knowledge about who knows what within a project team) and improving the team’s coordinating ability based on it.

However, transactive memory and expertise coordination ability do not seem to fully explain the nuanced effect of shared experience on team performance. In particular, the literature finds mixed results on the effect of expertise coordination on team performance. Some studies evidence that expertise coordination ability improves team performance (e.g., Faraj and Sproull 2000), while other studies show that the relationship between expertise coordination and team performance varies depending upon other factors,
such as the characteristics of task and team environments (Levesque et al. 2001; Majchrzak et al. 2007; Van Der Vegt and Bunderson 2005). Although transactive memory and expertise coordination ability emphasize knowledge about team member’s task expertise and a collaborative division of labor based on members’ expertise, they do not fully consider the power of common thinking, perception, belief, and expectation on critical components of team and task environments shared by team members, which is also known to influence team performance (Ellis 2006).

For this reason, in this paper, we use the concept of shared mental models (Cannon-Bowers et al. 1993; Duncan et al. 1996; Klimoski and Mohammed 1994; Mohammed and Dumville 2001; Rouse et al. 1992) as an underlying theoretical mechanism to explain the nuanced relationship between team members’ prior shared experience and team performance, especially how the relationship is moderated by the outcome of shared experience. Although there are numerous conceptualizations and definitions, shared mental models are generally defined as “team members’ shared, organized understanding and mental representation of knowledge about the key elements of the team’s relevant environment” (Mohammed and Dumville 2001). Shared mental models allow team members to understand the nature of task and recognize the behaviors of other teammates as well as the relationships among the components of task and team environments (Cannon-Bowers et al. 1993; Duncan et al. 1996; Rouse et al. 1992). Shared mental models provide team members with a common knowledge on a given task and enable them to predict the unforeseen problems of the task, expect what other team members will behave, quickly tailor their expectations and activities, and effectively solve the problems (Cannon-Bowers et al. 1993; Rouse et al. 1992). In other words, shared mental models help the team describe, explain, and predict problems, generate alternative solution, and find the best solutions for the problems (Klimoski and Mohammed 1994; Mathieu et al. 2000; Rouse et al. 1992; Waller et al. 2004). The literature find that shared mental models are positively related to team performance (Duncan et al. 1996; Mathieu et al. 2000; Rouse et al. 1992).

Shared mental models are different from transactive memory and expertise coordination ability. One critical difference is that transactive memory and expert coordination ability focus on the differences among team members and emphasizes a differentiation function of team members, while shared mental models are based on team members’ common perception, belief, and expectation and emphasizes an integrative function (Ellis 2006). Another difference is its contents. Shared mental models consist of two parts: task-work and task-work models (Cannon-Bowers et al. 1993). Task-work models refer to an understanding of activities and processes which have to be carried to perform the team task, while team-work models include conceptualization of the need for communication, compensation, mutual monitoring, and internal coordination which generally needed for the team to function effectively (Cannon-Bowers et al. 1993). Although transactive memory and expertise coordination ability have some components of team-work models, such as coordination strategy, they mainly relay on task-work models. This paper focuses more on the shared mental models’ integrative function and the team-work knowledge. More specifically, we argue how the different outcomes of shared experience affect the process of building and updating team-work shared mental models and team performance.

**Knowledge transfer and integration to build shared mental models**

Knowledge transfer and knowledge integration have been used to explain the link between individual experiences and team performance (Argote 2012; Jerez-Gomez et al. 2005). Knowledge transfer refers to the internal spreading process of individual team member’s knowledge to other team members, while knowledge integration refers to the process of synthesizing a corpus of knowledge and developing action plans for a given task through discussions and debates among team members (Argote 2012; Huber 1991; Walsh and Ungson 1991). A team’s shared mental models are built and continuously updated by these two processes.

Knowledge transfer within a team plays a significant role in building a team’s shared mental models (Middleton and Brown 2005; Reese and Fivush 2008). It leads to mutual understanding among team members and provides the team with a common knowledge foundation (Van den Bossche et al. 2011). Some researchers emphasize the role of “held in common” in building shared mental models (Argote 2012; Weick and Roberts 1993). The common things being held among team members include not only their knowledge on tasks and work processes (Wegner 1987), but also their thoughts, beliefs, perceptions, and feelings on task and team environments that leads to convergent common expectations about team members’ behaviors and interactions within the team (Monteverde 1995).
After transferring knowledge, the team takes a further step to reinforce its shared mental models. The team integrates individuals’ knowledge to synthesize new knowledge and develop appropriate action plans for the task given to the team (Huber 1991; Walsh and Ungson 1991). This knowledge integration process leads to an agreed-upon interpretation among team members about team and task environments (Van den Bossche et al. 2011) and increases the team’s absorptive capacity to explore alternative action plans and select the best one by identifying, assimilating, and exploiting knowledge (Cohen and Levinthal 1990). Knowledge integration creates a collective corpus of knowledge with new and existing experiences and make alternative action plans to solve the given new problem (Huber 1991; Kolb and Kolb 2005; Walsh and Ungson 1991). Knowledge integration also helps the team establish common expectations about team members’ behaviors and interactions, and based on them, the team is able to efficiently determine the work strategies, work priority, and work procedures to undertake the task in a complicated project situation (Klimoski and Mohammed 1994; Monteverde 1995).

Sharing relevant knowledge via knowledge transfer is necessary in order to build shared mental models, but it is not sufficient. Knowledge shared among team members must be contextualized to the given task environment, through knowledge integration, in order to apply the knowledge to the task (Haas and Hansen 2005; Pfeffer and Sutton 1999). If team members do not appropriately understand the nature of the task, work processes, and the roles and relationships among team members, they are not able to contribute effectively and align their behaviors to the needs of the team. Thus, good shared mental models are built only when both knowledge transfer and integration processes are effective.

Knowledge transfer and integration is not a one-time process. A team keeps updating shared mental models through continuous knowledge transfer and integration processes. These continuous processes provide a team with a double-loop learning environment for developing action plans, based on the shared mental models reinforced within the team (Argyris 1976; Argyris and Schön 1978). This double-loop learning provides the team with trial and error opportunities to look for other innovations and new routines and try them to modify existing shared mental models for achieving better performance.

**Role of communication in knowledge transfer and integration**

Communications are known to form a cognitive consensus and shared memory among people (Coman et al. 2009; Cuc et al. 2007). For example, speaking and listening activities trigger memory retrieval and store. Speakers retrieve information from memory, and listeners store information into memory. If speakers repeat something to listeners, the related information is reinforced in both parties’ memories and subsequently remembered better. On the other hand, if something is unmentioned in communications, it is less likely to be stored and retrieved later. For example, memories of traumatic events are less likely to be communicated among team members, and they are hardly be stored in the team’s shared mental models (James et al. 2016).

Communications among team members play a pivotal role in knowledge transfer and integration, and they have a significant impact on building shared mental models (Middleton and Brown 2005; Reese and Fivush 2008). In general, people in the same team talk to each other about past experience or knowledge and share their feeling, thoughts, ideas, and belief on common subjects. They sometimes have more formal, extensive communications, such as discussions, debates, and team meetings, in order to resolve conflicts, make decisions, and find solutions for identified problems (Bradley et al. 2015; Daniels and Walker 2001). Communications among team members serve a good means for collaborative learning and conflict resolution, and they often produce better team performance (Daniels and Walker 2001).

Although both knowledge transfer and integration processes are enabled by communications among team members, they need different types and depths of communications. Knowledge transfer can be triggered by basic communications, such as informal chats and conversations, while knowledge integration requires intensive communications, such as discussions and debates. For instance, team members may chat about their prior experience and thoughts with each other, which enables the team to possess the same set of memories about individual’s knowledge and thoughts. However, by chatting, the team may not be able to agree on some knowledge and thoughts, if they are different or conflicting among members, and may not be able to synthesize new knowledge or find appropriate action plans for the given task. This implies that a team’s shared mental models are differentially influenced by the type and depth of communications occurring within the team.
Shared experience generally promotes active conversations among team members. Shared experience encourages group communications and individual expressions (Arrow and McGrath 1995) because interpersonal knowledge and attraction reduce anxiety (Gruenfeld et al. 1996) and the fear of peer judgment (Larey and Paulus 1999). Formal or informal conversations among those who have worked together before provide a comfortable atmosphere where they can deeply share thoughts and feelings toward work strategies. With active conversations among team members, a team better understands the nature of team tasks. However, every shared experience does not provide the same depth and the same type of communications among team members. Some experience may offer more effective and deep communication, while some other experience may provide only certain type of communications which may limit effective knowledge transfer and integration among team members.

**Outcome of shared experience, communications, knowledge transfer and integration, shared mental models, and team performance**

We have discussed so far that shaped experience among team members is converted to team performance gains through effective shared mental models of the project team. A team's shared mental models are built and reshaped by knowledge transfer and integration processes which are influenced by communications among team members. In this section, we explain that how various outcomes of shared experience differentially affect the communications among project team members, knowledge transfer and integration processes, building and updating shared mental models, and team performance.

The organization learning literature finds that teams learn differently from their prior experience depending on the outcomes of experiences (Cyert and March 1963; Huber 1991; Levitt and March 1988). Teams evaluate the values of prior experience and select different lessons to be remembered and utilized. This suggests that teams may not react to every shared experience in the same way, too. Depending on the outcome of shared experience (i.e., whether a prior project where team members worked together was big success, big failure, or neither of both), different types and depths of communications among team members may emerge, which may differentially affect the building processes of shared mental models and team performance.

Following the related prior studies (Haunschild and Sullivan 2002; Kim and Miner 2007; Sitkin 1992), we conceptualize success and failure as a continuous measure rather than a distinct dichotomy because two success (or failure) outcomes may have different degrees of success (or failure). We classify the degree of failure (or success) of the outcome of shared experience into three relative levels, namely, big failure, big success, and a middle between these two extremes, referred to as small success or failure.

Small success or failure offers new knowledge, and this new knowledge builds and reshapes shared mental models (Sitkin 1992). Small success fosters subsequent performance by facilitating the refinement of successful routines in general, while organizations often encourage experimentation and learning from small failure (Baum and Dahlin 2007; Chuang and Baum 2003; Haunschild and Sullivan 2002; Kim and Miner 2007; Li and Rajagopalan 1997; Madsen and Desai 2010; Sitkin 1992). A team that experiences small failure is likely to establish investigative bodies to uncover the causes and lessons of the failure (Cannon and Edmondson 2005). Failure is considered as a learning opportunity. With findings and lessons from small failure, a team tends to try self-correction in order to perform better in subsequent tasks (Sitkin 1992). Failure contains richer cues explaining causality and entails a positive learning potential (Baum and Dahlin 2007; Madsen and Desai 2010).

Through the process of knowledge transfer, lessons learned from small success or failure are shared with team members, and through the process of knowledge integration, new lessons are integrated with existing shared mental models (Levitt and March 1988; Nelson and Winter 2009). Small failure sometimes may bring some conflicts and debates among team members, but they are generally constructive ones which help the team find a cause of the failure and a remedy for the cause (Bradley et al. 2015; Jehn et al. 2008). Constructive debates are a good means for collaborative learning and conflict resolution because they invoke more information search and deliberate considerations (Daniels and Walker 2001). Having constructive debates, a team can judge good items and bad items for their situation and decide a better solution by comparing and prioritizing them in the process of knowledge integration. Small success or failure may trigger changes in beliefs and values among team members and refine existing routines or work processes (Argyris and Schön 1978; Cope and Watts 2000). However, integration with new lessons gained
from small success or failure and existing shared mental models generally offers better solutions for forthcoming tasks.

That being said, every failure or success does not provide a positive impact on building or maintaining effective shared mental models (Kim and Miner 2007; Sitkin 1992). When success or failure is significantly big, this positive impact is like to diminish and a negative impact may arise (Kim and Miner 2007). Big failure shared experience makes knowledge transfer process less effective compared to small failure. Team members, who have experienced a big failure together, tend to avoid talking about their failure with each other and also to the other members in the same team (James et al. 2016). Big failure also makes a team hide disfavored information of failure so that people in the team fail to search and retrieve related information and loose a chance to utilize lessons which may be learned from big failure. The lack of communication and hidden information make knowledge transfer less effective compared to small success or failure (Argote et al. 1990; Darr et al. 1995). Big failure also hinders effective knowledge integration, either. People, who suffered traumatic failures, tend to think their existing mental models are inadequate and abandon them, even if this is not the case, without an appropriate post mortem process (Cyert and March 1963; March 1981). They often just seek new ideas and lose a chance to reinforce or correctly update their existing mental models. Big failure may destroy the trust among team members (Jones and George 1998). Instead of properly investigating the cause of failure, people who have experienced big failure together tend to blame each other and withdraw trust on the others. A crack on the trust among team members negatively affects the communications among them. Hence, knowledge integration process becomes less effective when a team shares big failure rather than small success or failure.

Big success shared experience hampers building effective mental models by limiting knowledge transfer and knowledge integration. When members in a team have experienced big success together, they tend to interpret the success as an evidence that their existing mental models represent the world well and further development of the models is unnecessary (Lant 1992; March and Shapira 1992; Ross and Sicoly 1979). They often become too overconfident about the adequacy of their knowledge gained from big success (Louis and Sutton 1991). They do not listen to other people, including the other team members who have not experienced the big success together, and force the other members to follow their belief and ways without proper discussions or debates. They often tend to ignore information about the world and do not make any further refinement on the existing mental models (Audia et al. 2000; Hayward et al. 2004). They refuse to learn by not searching for alternatives for new tasks that may be different from the prior task of success. The lack of communication among team members and limited access to new information makes knowledge transfer and integration process, which is essential to update shared mental models for upcoming new tasks, less effective, compared to the case of small success or failure shared experience.

Based on the above arguments, we expect that the effect of shared experience among team members on project team performance is nonlinearly affected by the degree of failure (or success) of shared experience. The effect of shared experience would be minimized when team members experienced big success or big failure together, while the effect of shared experience would be maximized when team members experienced small success or failure together. Thus, we hypothesize:

**Hypothesis 1:** The outcome of shared experience has a nonlinear moderating effect on the association between shared experience and software project performance in such a way that the association is strongest under intermediate degrees of failure (or success) of shared experience but comparatively weaker when the degree of failure (or success) of shared experience is high or low.

**Outcome of shared experience and depreciation of knowledge gained from shared experience**

Knowledge cumulated from prior experience does not persist forever. In other words, the effect of prior experience diminishes over time. This phenomenon is generally referred to as knowledge depreciation in the literature (Argote 2012; Darr et al. 1995). Knowledge depreciation has been mainly observed with task experience repeating the same or related tasks (Argote 2012; Darr et al. 1995; Ebbinghaus 1964; Thompson 2007). Shared experience is no different. Kang and Hur (2013) find that the effect of shared experience on team performance depreciates over time like task experience. We extend our understanding on the depreciation of knowledge gained from shared experience to consider that the outcome of shared experience may moderate the depreciation of knowledge.
One of the main causes of knowledge depreciation is time (Argote 2012; Darr et al. 1995). Knowledge decays over time (Ebbinghaus 1964). Both people and organizations remember more recent experience than old experience (Casey 1997). New experience may block the retrieval of past lessons learned, while past experience not mentioned may be forgotten (Underwood 1957). Knowledge depreciation is accelerated when there is an interruption or a prolonged break in the learning process (Argote 2012); however, knowledge depreciates over time even in the absence of a break or interruption (Thompson 2007). Team members continuously transfer and integrate knowledge to build and maintain their shared mental models. During this continuous process, knowledge, gained from old experience and not frequently retrieved, may be phased out from the mental models or replaced with new knowledge gained from recent experience (Casey and Olivera 2011; Feldman 2000). Hence, even if team members’ cumulative shared experience help to build effective shared mental models, knowledge from old shared experience is less likely to be remembered than one from recent shared experience (Casey 1997). Therefore, the effect of shared experience is expected to diminish as the lapse of shared experience (i.e., the time interval between the focal project and the prior project where project team members worked together) gets increased (Kang and Hur 2013).

However, with respect to shared experience, knowledge depreciation seems to be also influenced by another significant factor, that is, the outcome of shared experience. More specifically, different types and depths of communication among team members, caused by the different outcomes of shared experience, differentially moderate the process of knowledge depreciation. Another cause to explain knowledge depreciation is intentional forgetting (or not remembering). Intentional forgetting is also referred to as strategic forgetting or organized forgetting in the literature (De Holan and Phillips 2004; Nissley and Casey 2002). Intentional forgetting has a substitutive relationship with knowledge depreciation (Argote and Ingram 2000; James et al. 2016; Walsh and Ungson 1991), while intentional remembering has an interference relationship with knowledge depreciation (Boone et al. 2008). Aforementioned, team members sharing prior big failure (or big success) are not likely to communicate with each other about the experience, because they do not want to remember or they become too overconfident to believe there is nothing new to remember and fail to integrate new knowledge with their existing shared mental models (De Holan and Phillips 2004). Hence, the amount of knowledge newly added to shared mental models from big failure or success shared experience is limited compared to that from small failure or success shared experience. Even if new knowledge is gained from big failure or success shared experience, it is less likely to be codified and embedded in shared mental models. This means that shared experience in big failure or success is already forgotten or not remembered intentionally which leaves almost nothing in shared mental models to depreciate over time (De Holan and Phillips 2004; Nissley and Casey 2002). This implies that, when it comes to big failure or big success shared experience, intentional forgetting and the time lapse of shared experience have a substitutive relationship (Argote and Ingram 2000; Walsh and Ungson 1991) – both time lapse and intentional forgetting try to depreciate knowledge gained from shared experience, but they do redundantly. Given that intentional forgetting occurs first, it substitutes for the depreciation of knowledge caused by the time lapse of shared experience so that the depreciation effect of knowledge caused by time lapse becomes relatively small. On the other hand, when it comes to small failure or success shared experience, this substitutive effect does not exist or is very limited so that the depreciation effect of knowledge gained from shared experience caused by time lapse remain relatively large. Thus, we hypothesize:

**Hypothesis 2:** The lapse of shared experience has a nonlinear moderating effect on the effect of shared experience on project performance in such a way that the inverse U-shape association between shared experience and project performance becomes more flattened as the lapse of shared experience increases.

**Study Method**

**Data**

We have collected and analyzed an extensive archival data set from a software service company. The company provides contract-based custom software development for a variety of applications to a broad range of industries.

Our data set contains detailed information on software development projects (e.g., project actual schedule, plan, actual cost, budget, profit, customer, industry, etc.) that ended in between 2005 and 2007. The
average project duration is 10 months and the average project team size is 10. We also had access to the company’s entire records of employee’s project assignment (i.e., which projects each employee worked on) dating back to the company’s founding in the late 1980s. This data was used to operationalize the project team members’ shared experience.

**Dependent Variable**

**Project Performance.** We measure the performance of a software development project using the project’s actual labor effort in person-month. This effort is controlled for by the project size, project duration, and other factors in our regression analysis (see Control Variables section). Therefore, the effect of shared experience represents the reduction of effort. The effort measure has been widely used in other software development studies, especially those that examine the effects of prior experience, including shared experience (Boh et al. 2007; Espinosa et al. 2007; Narayanan et al. 2009; Subramanyam et al. 2012).

Project effort may have tradeoffs with project quality. For example, less project effort can be achieved at the cost of project quality (e.g., completing a software development project with less effort but many defects), and vice versa. For this reason, quality has to be controlled for measuring project effect more accurately (Espinosa et al. 2007). The company that we study strictly controls the quality of software projects. In line with standard statistical testing methods used in practice (Singpurwalla and Wilson 2012), the company adheres to the following final release testing procedure: The company has an independent quality assurance (QA) team consisting of software test specialists. For each project, this QA team conducts the final release tests. The QA team first creates an extensive set of test cases and test values from project documents. These test cases and values are very extensive and thorough because the company cares deeply about the quality of the delivered software systems. Although the number of test cases and values vary by project size, the release test usually takes at least a couple of days and sometimes even more than a week. If a system does not produce any errors in the defined test cases and values, the system is said to have passed the final release test. If the final release test identifies errors in the prepared set of test cases and values, the system is considered to have failed the test. In this case, the project team has to investigate the whole system again against the functional and non-functional requirements and fix all the errors found in the release test, and the QA team must repeat the full release test using a different (but still extensive) set of test cases and values. The release test procedures are repeated until the system passes the test. Therefore, our project effort measure is error free (i.e., without any delivery defect), similar to the ones used in Espinosa et al. (2007). It measures total development effort of a software project until the project delivers the promised system without any error to the customer.

Our effort deviation data is skewed; therefore, a natural logarithm transformation of the effort data is employed in our analysis like other studies (Boh et al. 2007; Espinosa et al. 2007; Narayanan et al. 2009; Subramanyam et al. 2012). Also, we reverse the sign of this variable, by multiplying -1, to make it easier to interpret the regression results similar to Espinosa et al. (2007).

**Independent and Moderator Variables**

**Shared Experience.** We define a measure of shared experience (SharedExp) used by a number of prior studies (Boh et al. 2007; Espinosa et al. 2007; Huckman et al. 2009; Staats 2012). Shared experience is computed as follows:

\[
\text{SharedExp} = \frac{\sum_{d=1}^{N} \sum_{i=1}^{N} w_{ij}}{N(N-1)/2}
\]

where \(w_{ij}\) is the number of prior projects in which team members \(i\) and \(j\) worked together before, and \(N\) is the number of members in the focal project team. The larger shared experience value, the more prior projects in which project team members worked together before.

**Failure Index of Shared Experience.** In order to determine the extent to which a prior project where project members have worked on together is unsuccessful (or successful), we evaluate whether the prior project is completed on time (schedule index), within budget (cost index), and profitable (profit index) following the conventional notions to represent the failure of software projects in the literature (Brooks 1995; El Emam and Koru 2008; Ewusi-Mensah 2003). We define the value of the indices as follows: If the project is completed earlier than its planned due date, the schedule index of the project is -1; if the project...
Shared Experience, Outcome, and Forgetting

is completed exactly on its planned due date, the schedule index is 0; if the project is completed after the planned due date, the schedule index is 1. Similarly, if the project is completed by spending less than its planned cost budget, the cost index of the project is -1; if the project spends just all its planned budget, the cost index is 0; if the project spends more than its planned budget, the cost index is 1. Lastly, if the project makes a profit, the profit index of the project is -1; if the project does not make a profit or loss, the profit index is 0; if the project makes a loss, the profit index is 1. The overall failure index of project $k$ ($f_k$) is the average of the above three indices: (schedule index + cost index + profit index)/3. Then, the failure index of shared experience (FailureIndexSharedExp) is computed as follows:

$$\text{FailureIndexOfSharedExp} = \frac{\sum_{i=1}^{N-1} \sum_{j=1}^{N} \sum_{k=1}^{w_{ijk}} f_{ijk}}{\sum_{i=1}^{N-1} \sum_{j=1}^{N} l_{ijk}}$$

where $f_{ijk}$ is the failure index of the $k$th prior project in which team members $i$ and $j$ worked together before. A larger failure index means that the project team’s shared experience is more associated with unsuccessful prior projects than successful projects. This failure index variable represents the degree of failure (or success) of shared experience and is used to test Hypothesis 1.

Lapse of Shared Experience. The lapse of shared experience is defined as the time interval (in days) between the prior project with which shared experience is associated and the focal project. To measure the lapse of shared experience, we calculate the average lapse across all team members’ shared experiences in the project team (LapseOfSharedExp):

$$\text{LapseOfSharedExp} = \frac{\sum_{i=1}^{N-1} \sum_{j=1}^{N} \sum_{k=1}^{w_{ijk}} l_{ijk}}{\sum_{i=1}^{N-1} \sum_{j=1}^{N} w_{ijk}}$$

where $l_{ijk}$ is the time interval between the end date of the $k$th prior project in which team members $i$ and $j$ worked together before and the beginning of the focal project. The smaller lapse, the more recent shared experience. This lapse variable is used to test Hypothesis 2.

The following stylized example illustrates how these three variables are computed. Suppose that M1, M2, and M3 are project team members. M1 and M3 previously worked together once in project P3 which was completed 90 days ago, and M2 and M3 worked together twice in projects P1 and P2 which were completed 60 and 30 days ago, respectively. Then shared experience of the focal project team (SharedExp) becomes 1 ($=(1+2)/((3\times2)/2)$) and the lapse of shared experience (LapseOfSharedExp) becomes 60 ($=(90+60+30)/3$) days. Suppose that P3 was successful in every aspect. It was completed earlier than its plan (schedule index=−1), spent less cost than its planned budget (cost index=−1), and generated a profit (profit index=−1). Hence, the failure index of the shared experience between M1 and M2 in P3 is −1 (=$-((-1)+(-1)+(-1))/3$). P1 was successful in some aspects but neutral or unsuccessful in some others. It was delayed (schedule index=−1), while it spent cost as planned (cost index=0) and made a profit (profit index=−1). Hence the failure index of the shared experience between M2 and M3 in P1 is 0 (=$(1+0+(-1))/3$). P2 was unsuccessful in every aspect. It was delayed (schedule index=−1), spent more cost than its plan (cost index=−1), and generated a loss (profit index=−1). Hence, the failure index of the shared experience between M2 and M3 in P2 is 1 (=$1+1+1)/3$). Therefore, the average failure index of shared experience of the focal project team (FailureIndexOfSharedExp) is 0 (=$(1+0+(-1))/3$).

Control Variables

Project Size. We use the project’s estimated effort (EstimatedEffort) and duration (EstimatedDuration) to control for the project size (Huckman et al. 2009). Since the estimated effort and duration is determined based on the estimated project size, they are appropriate proxies for the latter. In order to avoid potential endogeneity problems, we use the estimated project effort and duration (Huckman et al. 2009).

Individual Experience. Individual project members’ prior project experience are known to affect project team performance (Argote 2012). We compute the total number of prior projects which an individual member experienced before and use it as the individual’s prior project experience. We use the average of project team members’ individual experience for the project team (IndividualExp).

Software Process Type. In software development, different types of software processes are used, and they may affect software project performance. The company where we collected the data uses two distinct

Thirty Eighth International Conference on Information Systems, South Korea 2017 9
types of software processes. One is based on traditional waterfall and the other is iterative approach. We use a binary variable (SoftwareProcessType) to represent whether the project’s software process is based on traditional waterfall (0) or iterations (1).

**Customer Industry.** The unique characteristics of the customer industry may affect software development performance (Ethiraj et al. 2005). We include industry sector dummies for public (IndustrySector\textsubscript{public}), manufacturing/production (IndustrySector\textsubscript{mfg}), and financial sectors (IndustrySector\textsubscript{finance}), with the service sector as the base case.

**Years.** We include year dummies for the project end year (Year\textsubscript{2005}, Year\textsubscript{2006}) with 2007 as the baseline, to control for macroeconomic conditions such as inflation or business cycle and technological progress which may affect project performance (Argote 2012; Boh et al. 2007).

**Team Size.** We use the number of project team members (TeamSize) to control for the effect of project team size (Huckman et al. 2009). Larger project teams may suffer from coordination and communication challenges resulting in decreased performance (Brooks 1995; Ethiraj et al. 2005).

The descriptive statistics and inter-correlations of all variables are shown in Table 1.

**Table 1. Descriptive Statistics and Inter-correlations**

<table>
<thead>
<tr>
<th>Variables</th>
<th>Mean</th>
<th>SD</th>
<th>Min</th>
<th>Max</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ProjectPerformance</td>
<td>-3.24</td>
<td>1.29</td>
<td>-6.92</td>
<td>0.92</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. EstimatedEffort</td>
<td>58.41</td>
<td>105.43</td>
<td>0.00</td>
<td>923.00</td>
<td>-2.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. EstimatedDuration</td>
<td>9.79</td>
<td>9.03</td>
<td>0.37</td>
<td>75.27</td>
<td>-0.50</td>
<td>0.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. IndividualExp</td>
<td>10.23</td>
<td>6.53</td>
<td>0.00</td>
<td>43.00</td>
<td>0.30</td>
<td>-0.20</td>
<td>-0.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. SoftwareProcessType</td>
<td>0.47</td>
<td>0.50</td>
<td>0.00</td>
<td>1.00</td>
<td>-0.22</td>
<td>0.12</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. IndustrySector\textsubscript{public}</td>
<td>0.27</td>
<td>0.45</td>
<td>0.00</td>
<td>1.00</td>
<td>-0.25</td>
<td>0.15</td>
<td>0.13</td>
<td>0.09</td>
<td>0.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. IndustrySector\textsubscript{mfg}</td>
<td>0.16</td>
<td>0.37</td>
<td>0.00</td>
<td>1.00</td>
<td>-0.10</td>
<td>0.11</td>
<td>0.05</td>
<td>0.17</td>
<td>0.18</td>
<td>0.27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. IndustrySector\textsubscript{finance}</td>
<td>0.14</td>
<td>0.34</td>
<td>0.00</td>
<td>1.00</td>
<td>0.01</td>
<td>0.03</td>
<td>0.12</td>
<td>0.06</td>
<td>0.14</td>
<td>0.24</td>
<td>0.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Year\textsubscript{2005}</td>
<td>0.37</td>
<td>0.48</td>
<td>0.00</td>
<td>1.00</td>
<td>0.06</td>
<td>-0.07</td>
<td>0.02</td>
<td>-0.26</td>
<td>-0.02</td>
<td>-0.07</td>
<td>0.03</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. Year\textsubscript{2006}</td>
<td>0.35</td>
<td>0.48</td>
<td>0.00</td>
<td>1.00</td>
<td>0.04</td>
<td>0.03</td>
<td>-0.19</td>
<td>0.10</td>
<td>0.17</td>
<td>0.04</td>
<td>0.04</td>
<td>0.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Team Size</td>
<td>9.26</td>
<td>10.07</td>
<td>2.00</td>
<td>89.00</td>
<td>-0.72</td>
<td>0.82</td>
<td>0.33</td>
<td>-0.22</td>
<td>0.19</td>
<td>0.10</td>
<td>0.17</td>
<td>0.04</td>
<td>-0.04</td>
<td>-0.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. SharedExp</td>
<td>1.17</td>
<td>2.21</td>
<td>0.00</td>
<td>16.00</td>
<td>0.30</td>
<td>-0.17</td>
<td>-0.20</td>
<td>0.58</td>
<td>-0.02</td>
<td>-0.06</td>
<td>0.02</td>
<td>-0.25</td>
<td>0.13</td>
<td>-0.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. FailureIndexOfSharedExp</td>
<td>0.14</td>
<td>0.20</td>
<td>0.00</td>
<td>1.00</td>
<td>-0.10</td>
<td>0.05</td>
<td>0.11</td>
<td>0.02</td>
<td>0.18</td>
<td>0.01</td>
<td>0.01</td>
<td>0.12</td>
<td>0.07</td>
<td>0.10</td>
<td>0.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14. LapseOfSharedExp</td>
<td>179.49</td>
<td>184.60</td>
<td>0.00</td>
<td>885.00</td>
<td>-0.05</td>
<td>0.03</td>
<td>0.06</td>
<td>0.24</td>
<td>-0.01</td>
<td>0.02</td>
<td>0.08</td>
<td>-0.07</td>
<td>-0.38</td>
<td>0.10</td>
<td>0.10</td>
<td>0.30</td>
<td>0.32</td>
</tr>
</tbody>
</table>

**Notes:** Bold denotes significance at $p<0.05$.

**Analytical Approach**

The proposed hypotheses are the nonlinear moderating effects of the failure index of shared experience and the lapse of shared experience. The most common approach to test the nonlinear moderating effects is the regression model with the interactions between the square products of moderators and independent variables (Cadogan et al. 2009; Gefen and Pavlou 2012; Schilke 2014). Following this convention, we define the following regression equation to test our hypotheses:

$$
\beta_0 + \beta_1 \times \text{EstimatedEffort} + \beta_2 \times \text{EstimatedDuration} + \beta_3 \times \text{IndividualExp} + \\
\beta_4 \times \text{SoftwareProcessType} + \beta_5 \times \text{IndustrySector}_{\text{public}} + \beta_6 \times \text{IndustrySector}_{\text{mfg}} + \\
\beta_7 \times \text{IndustrySector}_{\text{finance}} + \beta_8 \times \text{Year}_{2005} + \beta_9 \times \text{Year}_{2006} + \beta_{10} \times \text{TeamSize} + \beta_{11} \times \text{SharedExp} + \\
\beta_{12} \times \text{FailureIndexOfSharedExp} + \beta_{13} \times \text{SharedExp} \times \text{FailureIndexOfSharedExp} + \\
\beta_{14} \times \text{FailureIndexOfSharedExp}^2 + \beta_{15} \times \text{SharedExp} \times \text{FailureIndexOfSharedExp}^2 + \\
\beta_{16} \times \text{LapseOfSharedExp} + \beta_{17} \times \text{SharedExp} \times \text{LapseOfSharedExp} + \\
\beta_{18} \times \text{FailureIndexOfSharedExp} \times \text{LapseOfSharedExp} + \\
\beta_{19} \times \text{SharedExp} \times \text{FailureIndexOfSharedExp} \times \text{LapseOfSharedExp} + \\
\beta_{20} \times \text{LapseOfSharedExp} \times \text{FailureIndexOfSharedExp} \times \text{LapseOfSharedExp} + \\
\beta_{21} \times \text{SharedExp} \times \text{LapseOfSharedExp} \times \text{FailureIndexOfSharedExp} \times \text{LapseOfSharedExp}.
$$
A significant coefficient of the squared moderator product term (SharedExp × FailureIndexOfSharedExp²) would indicate the existence of quadratic moderation effect. Hence, β_{22} is used to test Hypothesis 1. A significant, negative β_{22} would suggest that shared experience has an inverse U-shaped effect on project performance with respect to the failure index of shared experience. Similarly, a significant coefficient of the interaction term between the quadratic moderator product term and lapse moderator (SharedExp × FailureIndexOfSharedExp²) would indicate that the quadratic moderation is further moderated by the lapse of shared experience. β_{33} is used to test Hypothesis 2. A significant, positive β_{33} would suggest that the inverse U-shaped effect of shaped experience on project performance in the failure index becomes flattened as the lapse of shared experience increases.

### Results and Discussion

#### Results

The analysis is conducted using a hierarchical approach according to the standard practice for analyzing models with interaction terms (Aiken et al. 1991). We first estimate a baseline model with all control variables as well as the shared experience variable (Model 1). We then add the failure index of shared experience variable and the interaction terms, including the squared moderator product term, with the shared experience variable (Model 2). Model 2 is used to test Hypothesis 1. Lastly, we add the lapse of shared experience variable and its interaction terms with the shared experience and the failure index variables (Model 3). Model 3 is used to test Hypothesis 2. This progressive approach allows us to check if the inclusion of these variables increases the explanatory power of the models (Aiken et al. 1991). The regression results are summarized in Table 2. The results show that the failure index of shared experience and its interaction terms add significant explanatory power to the model containing the shared experience and the other control variables (Model 1 vs. Model 2: ∆R²=0.0101, F=4.58, p<0.01). Similarly, the inclusion of the lapse of shared experience and its interaction terms also significantly increases the explanatory power of the model (Model 1 vs. Model 2: ∆R²=0.0091, F=5.13, p<0.01).

#### Table 2. Results – Project Performance

<table>
<thead>
<tr>
<th>Variables</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-2.170***</td>
<td>-2.079***</td>
<td>-1.976***</td>
</tr>
<tr>
<td>EstimatedEffort</td>
<td>0.144</td>
<td>0.147</td>
<td>0.148</td>
</tr>
<tr>
<td>EstimatedDuration</td>
<td>0.001</td>
<td>0.007</td>
<td>0.002</td>
</tr>
<tr>
<td>IndividualExp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SoftwareProcessType</td>
<td>-0.201***</td>
<td>-0.193***</td>
<td>-0.213***</td>
</tr>
<tr>
<td>IndustrySector¹</td>
<td>0.070</td>
<td>0.085</td>
<td>0.085</td>
</tr>
<tr>
<td>IndustrySector²</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IndustrySector³</td>
<td>-0.478**</td>
<td>-0.448**</td>
<td>-0.424**</td>
</tr>
<tr>
<td>IndustrySector⁴</td>
<td>-0.244**</td>
<td>-0.228**</td>
<td>-0.235**</td>
</tr>
<tr>
<td>Year</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Year²</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Year³</td>
<td>-0.093</td>
<td>-0.092</td>
<td>-0.091</td>
</tr>
<tr>
<td>Team Size</td>
<td>-0.039**</td>
<td>-0.036**</td>
<td>-0.034**</td>
</tr>
<tr>
<td>SharedExp</td>
<td>0.087**</td>
<td>0.018</td>
<td>0.042</td>
</tr>
<tr>
<td>FailureIndexOfSharedExp</td>
<td>-1.459**</td>
<td>-1.365</td>
<td>-2.865**</td>
</tr>
<tr>
<td>FailureIndexOfSharedExp²</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SharedExp×FailureIndexOfSharedExp</td>
<td>0.470**</td>
<td>0.187</td>
<td>1.850**</td>
</tr>
<tr>
<td>FailureIndexOfSharedExp³</td>
<td>1.589**</td>
<td>0.536</td>
<td>3.291**</td>
</tr>
<tr>
<td>LapseOfSharedExp</td>
<td>-0.482</td>
<td>0.213</td>
<td>-2.237*</td>
</tr>
<tr>
<td>LapseOfSharedExp×SharedExp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>484</td>
<td>484</td>
<td>484</td>
</tr>
<tr>
<td>R²</td>
<td>0.6747</td>
<td>0.6844</td>
<td>0.6935</td>
</tr>
</tbody>
</table>

Thirty Eighth International Conference on Information Systems, South Korea 2017 11
Our results to be robust to the various dimensions. They are profit, schedule, and cost. We define the following five alternative failure index variables: profit, schedule, and cost. We find that all the five alternative variables provide the same hypotheses testing results: inverse U-shaped moderating effect of the failure index with shared experience and the flattening effect of the lapse of shared experience on the inverse U-shaped moderating effect. This result implies that our results to be robust to the various definitions for failure software development projects.

**Significance Levels:** *p<0.1, **p<0.05, ***p<0.01

<table>
<thead>
<tr>
<th>Adj R²</th>
<th>0.6671</th>
<th>0.6743</th>
<th>0.6796</th>
</tr>
</thead>
<tbody>
<tr>
<td>ΔR²</td>
<td>0.0101</td>
<td>0.0091</td>
<td></td>
</tr>
<tr>
<td>F(ΔR²)</td>
<td>4.58**</td>
<td>5.43***</td>
<td></td>
</tr>
</tbody>
</table>

We observe that the coefficients of explanatory variables remain relatively stable across all three models—an indicator that multicollinearity is not an issue. Nonetheless, we further check if our models suffer from a potential multicollinearity problem using the diagnostics based on condition index (CI) and variance inflation factors (VIF). They are found to be within recommended and acceptable limits (Max VIF=15.19, CI=10.80) (Belsley et al. 2005; Neter et al. 1996), suggesting that no problematic multicollinearity exists in our models.

In Model 1, we first find that our control variables are significantly associated with project performance. As expected, larger projects are associated with increased labor effort (decreasing the dependent variable) (EstimatedEffort: $\beta_1 = -0.004, p < 0.01$; EstimatedDuration: $\beta_2 = -0.026, p < 0.01$). Individual project experience increases project performance (IndividualExp: $\beta_3 = 0.001, p < 0.1$), which is consistent with the findings in the learning literature (Argote 2012). The development process type is found to affect project performance, suggesting that iterative processes may require more effort (SoftwareProcessType: $\beta_4 = -0.201, p < 0.01$) in our data set. The customer industries seem to have unique characteristics affecting software development performance (IndustrySector_public: $\beta_5 = -0.528, p < 0.01$; IndustrySector_mfg: $\beta_6 = -0.478, p < 0.01$; IndustrySector_finance: $\beta_7 = -0.244, p < 0.01$). On the other hand, the year dummies do not show any significant effect on project performance (Year2005: $\beta_8 = 0.080, ns$; Year2006: $\beta_9 = -0.093, ns$), suggesting that macroeconomic conditions do not change during 2005-2007 or even if they do so, they do not significantly affect project performance. Being consistent with the software complexity literature (Xia and Lee 2005), team size, a team coordination complexity measure, is found to negatively affect project performance (TeamSize: $\beta_{10} = -0.039, p < 0.01$). Finally, shared experience is found to increase project performance (SharedExp: $\beta_{11} = 0.087, p < 0.01$), which is consistent with the findings in the shared experience literature (Faraj and Sproull 2000; Wegner 1987).

Model 2 adds the failure index of shared experience and its interaction terms to test Hypothesis 1. We find that the interaction term between the shared experience and the squared product of the failure index variable has a negative, significant coefficient (SharedExp×FailureIndexOfSharedExp²: $\beta_{15} = -0.482, p < 0.05$), which suggests that the effect of shared experience on project performance has an inverse U-shape with respect to the failure index of shared experience. Therefore, Hypothesis 1 is supported.

Model 3 adds the lapse variable of shared experience and its interaction terms to test Hypothesis 2. We observe that the interaction term between the interaction term between the shared experience and the squared product of the failure index variables (SharedExp×FailureIndexOfSharedExp²) and the lapse variable (LapseOfSharedExp) has a positive, significant coefficient (SharedExp×FailureIndexOfSharedExp²×LapseOfSharedExp: $\beta_{21} = 0.066, p < 0.05$). This result implies that the negative coefficient of SharedExp×FailureIndexOfSharedExp² is increased as the lapse of shared experience increases. In other words, the inverse U-shaped effect of shared experience in the failure index becomes flattened toward zero as the lapse of shared experience increases. Hence, Hypothesis 2 is supported.

**Robustness Check**

We conduct additional analyses to check the robustness of the results. Our results may depend on the operationalization of the failure index of shared experience. Our failure index variable uses the three dimensions, schedule, cost, and profit all together. Although it represents a consolidated concept of failure of software projects, each dimension may have different moderating effects on the shared experience’s impact on project performance. Therefore, we define the following five alternative failure index variables using subsets of the three performance dimensions. They are [profit, schedule], [profit, cost], [profit], [schedule], and [cost]. We find that all the five alternative variables provide the same hypotheses testing results—inverse U-shaped moderating effect of the failure index with shared experience and the flattening effect of the lapse of shared experience on the inverse U-shaped moderating effect. This result implies that our results to be robust to the various definitions for failure software development projects.
Discussion of Results

We now further discuss the implications of the results by interpreting the regression coefficients. The nature of the inverse U-shaped effect of shared experience with respect to the failure index of shared experience (Hypothesis 1) is illustrated in Figure 1(a), and the nature of flattening moderation effect of the lapse of shared experience on the inverse U-shaped effect of shared experience on project performance (Hypothesis 2) is illustrated in Figure 1(b). When creating the graphs in Figures (a) and (b), the regression equation is evaluated at different levels of the failure index and lapse of shared experience, using the margins and marginsplot commands in STATA 13. In the both figures, the vertical axis of the graph represents values of regression coefficient for shared experience, and the horizontal axis of the graph represents values of the failure index of shared experience between two standard deviations below and above the mean (i.e., between -0.22 and 0.63). Two different values of the lapse of experience (low and high) are represented in black and gray colors (see the legend information of Figure 1(b)).

![Graphs showing association between project performance and shared experience](image)

**Figure 1. The association between project performance and shared experience as a function of the failure index and lapse of shared experience (with 95% confidence interval)**

The graph in Figure 1(a) represents the association between project performance and shared experience across different levels of the failure index of shared experience. The proposed inverse U-shaped relationship between project performance and shared experience across increasing values of the failure index of shared experience (Hypothesis 1) looks apparent in the graph. As shown in Figure 1(a), for project teams of which the failure index of shared experience is relatively low or high (i.e., team members worked together in highly successful or highly unsuccessful prior projects), the coefficient for the regression of project performance on shared experience is comparatively low (i.e., the effect of shared experience is low). On the other hand, at the moderate level of the failure index of shared experience, the coefficient becomes relatively high. According to the regression estimates (Model 2), the coefficient for the regression of project performance on shared experience is maximized to 0.16 when the failure index of shared experience is 0.49. In other words, one unit of shared experience would increase project performance by 0.16 when the failure index of shared experience is 0.49. One the other hand, when the failure index of project team’s shared experience is 0.28 (one standard deviation below 0.49), the increase in project performance by increasing one unit of shared experience would be reduced to 0.14, and when the failure index is 0.60 (one standard deviation above 0.49), then the increase would be reduced to 0.15, too.

Project performance is defined as a negative natural log of development effort. Therefore, the effects of shared experience and the moderating effect of the failure index of shared experience may be interpreted as reductions of development effort. According to the estimated regression coefficients, for a typical software project which involves 60 person-months of development effort (average value for the sample), if the project team had one more unit of shared experience, the project team could have saved 8.87 person-months in development effort when the failure index of project team’s total shared experience is 0.49. On the other hand, development effort that the project team could have saved would be reduced to 7.84 when the failure index is 0.28 (i.e., project team’s shared experience is involved with more successful prior
projects) and reduced to 8.36 person-months when the failure index is 0.60 (i.e., project team’s shared experience is involved with less successful prior projects), respectively.

In our regression results, the effect of shared experience is not always positive. Shared experience is found to significantly increase project performance only when shared experience is related to moderately successful or unsuccessful projects (i.e., when the failure index of shared experience is between 0.02 and 0.65 at \( p < 0.05 \)). Shared experience is found to significantly decrease project performance when shared experience is associated with highly successful prior projects (i.e., when the failure index is less than -0.37 at \( p < 0.05 \)).

Now we turn our attention to how the inverse U-shaped effect of shared experience is moderated by the lapse of shared experience. The two graphs in Figure 1(b) represent the association between project performance and shared experience across increasing values of the failure index of shared experience but for different levels of the lapse of shared experience. The black graph with circle connectors represents the association when the lapse of experience is low (160, a half of standard deviation below the mean), while the gray graph with rectangle connectors represents the association when the lapse of shared experience is high (332, a half of standard deviation above the mean). Consistent with the graph in Figure 1(a), the both graphs show inverse U-shapes; however, the black graph (i.e., low lapse) shows a steeper curvature than the gray graph (i.e., high lapse). The regression coefficient for \( SharedExp \times FailureIndex \times SharedExp^2 \) (\( SharedExp \) is the failure index of shared experience) represents the steepness of curvature of this quadratic inverse U-shape – the smaller negative value, the steeper curvature. According to our regression results (Model 3), the regression estimate for this coefficient is -1.28 for the black graph and -0.25 for the gray graph, respectively. These numbers also confirm that the black graph has a steeper curvature than the gray one.

When the failure index is 0.20 (median value of the sample), the estimated coefficient of shared experience becomes 0.15 for the black graph (lapse =160), and it becomes 0.12 for the gray graph (lapse=332). This means that when the shared experience’s lapse is 160, one unit of shared experience would increase project performance by 0.03 than when the lapse is 332. For a typical project that involves 60 person-months, the recent shared experience (lapse =160) would save 1.58 (=8.36-6.78) more person-months than the distant shared experience (lapse=332), given all else being equal. Not only the effect of shared experience on project performance but also the marginal change in the effect with respect to the failure index is affected by the lapse of shared experience. When the failure index of shared experience increases by 0.10 units and become 0.30, the estimated coefficient for shared experience is increased for both black and gray graphs because they are still on the left side of the inverse U-shape. However, the coefficient for black graph (lapse=160) would be increased by 0.05 (from 0.15 to 0.20), while the coefficient for the gray graph (lapse=332) would be increased by only 0.03 (from 0.12 to 0.15) because of the flattening moderating effect of the lapse of shared experience on the inverse U-shaped effect of shared experience on project performance. For the same reason, when the failure index is increased from 0.53 to 0.63 (two standard deviations above the mean), the coefficient for the black graph (lapse=160) would be decreased by 0.04 (from 0.21 to 0.17), while it would be decreased by just 0.01 (from 0.17 to 0.16) for the gray graph (lapse=332).

**Conclusion**

Due to a variety of software development, software development is generally conducted by temporal project teams rather than permanent teams. However, software development is a complex team task requiring extensive team coordination and collaboration (Faraj and Sproull 2000), which challenges contemporary software development practices relying on project teams consisting of temporal membership. For this reason, understanding how to increase project team performance has been a critical issue to software organizations for a long time. Shared experience (i.e., the experience of working together in the same prior projects) among project team members is one of the important factors determining software project performance and has been recently examined in some studies (Huckman et al. 2009; Staats 2012). Although these seminal studies make a significant contribution to understanding the basic role of shared experience in software development, the effect of shared experience seems to be much more complicated and vary by other environmental or contextual factors. In this study, we examine the two unexamined factors that moderate the effect of shared experience on team performance. They are the outcome of shared experience and the time lapse of shared experience.
Drawing upon the theory of shared mental models, we find the effect of shared experience on project team performance is moderated by the outcome of shared experience. More specifically, we find that the effect of shared experience shows an inverse U-shaped curve with respect to the degree of failure (or success) of the outcome of shared experience— the effect of shared experience is maximized when the outcome of shared experience is small success or failure, while the effect of shared experience is minimized when the outcome is big success or big failure.

In addition, we find that knowledge depreciation regarding shared experience differently occurs depending on the outcome of shared experience, too. Big success or big failure shared experience makes project team members intentionally forget (or not to remember) the experience and hinders them from storing the experience into shared mental models. This plays a substitutive role for knowledge depreciation caused by time lapse. We find that the depreciation of knowledge, gained from shared experience exhibits, caused by time lapse becomes weaker when the outcome of shared experience is big failure or big success, while the depreciation of knowledge caused by time lapse remain relatively large when the outcome is small success or failure. This effect is observed as a flattening effect on the inverse U-shaped curve effect of shared experience. As the lapse of shared experience gets larger (i.e., shared experience gets older), the inverse U-shaped curve becomes more flattened.

This study makes a theoretical contribution by finding two significant factors that moderate the effect of shared experience on project team performance. Moreover, the found effects are nonlinear moderations which are little studied in the literature. A nonlinear moderation provides more nuanced interpretations of a certain complex phenomenon which cannot be examined by standard linear moderation approaches (Schilke 2014).

This study also provides practical implications to software organizations. According to our results, composing a project team with members sharing prior big success or big failure experience may not be a good project staffing decision. If a project is found to be big success or big failure, separating the project team members out and assigning them to different subsequent projects rather than having the same team members in a subsequent project may minimize the negative impact of big success or big failure shared experience on subsequent projects.

Like other studies, this study has limitations some of which requires further research. First, our data is little outdated and the projects in our data set may not represent all types of software projects currently practiced. In particular, the projects in the data set do not include pure agile software projects which may require team coordination and collaboration more than traditional software projects. Although we believe our results are general findings which may be applicable to a variety of software development practices, we also acknowledge that the generalizability of our results may be limited. A replicating research with more recent software project data, including agile software projects, is one of the potential extensions of this study.

Another limitation is our data set is from one specific company. This focused research setting may enhance the internal validity of the results. However, it also implies that our findings may be influenced by the idiosyncrasies of the research site. Future research in other settings would further generalize and validate the results. Similarly, our study is based on empirical analysis on extensive archival project data. Although this empirical method provides rigorous results and nicely work for our research questions, it limits more nuanced discussions on what really happens in the team or team members. A mixed method combining the empirical method based on data and qualitative approach with interviews or observations would be an interesting extension of this study.

References


