7-1-2013

Knowledge Creation In Information Systems Development Teams: The Role Of Pair Programming And Peer Code Review

Kai Spohrer
University of Mannheim, Mannheim, Germany, spohrer@uni-mannheim.de

Thomas Kude
University of Mannheim, Mannheim, Germany, kude@uni-mannheim.de

Christoph T. Schmidt
SAP Research & Institute for Enterprise Systems, University of Mannheim, Walldorf, Germany, christoph.schmidt01@sap.com

Armin Heinzl
University of Mannheim, Mannheim, Germany, heinzl@uni-mannheim.de

KNOWLEDGE CREATION IN INFORMATION SYSTEMS DEVELOPMENT TEAMS: THE ROLE OF PAIR PROGRAMMING AND PEER CODE REVIEW

Kai Spohrer, University of Mannheim, L15, 1-6, 68161 Mannheim, Germany, spohrer@uni-mannheim.de

Thomas Kude, University of Mannheim, L15, 1-6, 68161 Mannheim, Germany, kude@uni-mannheim.de

Christoph T. Schmidt, SAP Research & Institute for Enterprise Systems, University of Mannheim, 69190 Walldorf, Germany christoph.schmidt01@sap.com

Armin Heinzl, University of Mannheim, L15, 1-6, 68161 Mannheim, Germany, heinzl@uni-mannheim.de

Abstract

Software developers increasingly rely on the four-eyes principle to ensure high quality early on in the development process. Implementing this principle, peer code review has recently been proposed as a less costly alternative to the more commonly used pair programming technique. This study examines the implications of these techniques on knowledge creation within information systems development teams. Thereby, it contrasts with prior research on pair programming and peer code review which has primarily focused on comparing direct quality gains to increased costs.

Based on a multiple-case study and guided by Nonaka's theory on organizational knowledge creation, this study improves our understanding of the varying effects of pair programming and peer code review, both on an individual and on a team level. Findings show that the techniques may be substitutes on a dyadic level, but may complement each other regarding their effects on team-level knowledge creation: if applied simultaneously, they may collectively enable socialization, externalization, combination, and internalization of knowledge. This study contributes not only to literature on development techniques by explaining the complementary nature of pair programming and peer code review, but also extends current literature by analyzing how development teams come to benefit from knowledge of their single members in daily work with these techniques.

Keywords: Pair programming, peer code review, knowledge creation, software development

1 Introduction

In order to ensure software quality early on in the IS development (ISD) process, programmers and ISD teams increasingly rely on pair programming (PP) (VersionOne, 2011). Indeed, introducing the four-eyes principle was frequently found to substantially improve code quality (Salleh et al., 2011). Recently, peer code review (PCR) has been suggested as a less costly alternative to reduce errors adhering to the four-eyes principle (Rigby et al., 2012). PCR originated from open-source software development, where PP is unfeasible due to the distributed nature of projects. Instead, open source developers often rely on PCR as a computer-mediated quality assurance technique that allows for validating software fragments submitted to the system by peers.

Proponents of PCR as an alternative to PP assume the two techniques to be functionally equivalent (Paulk, 2001). Given that both techniques enable the implementation of the four-eyes principle, pair programming and peer code review may indeed be substitutes concerning the common goal of software defect mitigation (cf. Müller, 2004, 2005; Rigby et al., 2012).

However, recent research suggests that beneficial effects of PP may go beyond solely reducing errors in the jointly developed code (Vidgen & Wang, 2009). In fact, PP was found to be most useful when developers are required to learn how to approach complex tasks (Balijepally et al., 2009; Williams & Kessler, 2000; Williams, 2000; Salleh et al., 2011). Moreover, there is initial evidence that PP may not only affect individual developers, but also facilitate the creation and exchange of knowledge within ISD teams (Vidgen & Wang, 2009; Rigby et al., 2012). While PCR may be a viable alternative to PP with regard to the underlying four-eyes principle, varying effects of the two techniques on team-level knowledge creation would cast doubt on their substitutive nature.

Consequently, the goal of this study is to examine implications of applying PP and PCR beyond direct code quality improvements. In particular, we aim at better understanding the theoretical mechanisms that link the utilization of PP and PCR on an individual or dyadic level with knowledge creation on a team level. In detail, we seek to answer the following question: *How do PP and PCR contribute to knowledge creation on a dyad and on a team level?*

In order to reach this goal, we conducted a multiple-case study of nine ISD teams within a large European software vendor. In line with the socio-technical nature of ISD, and particularly PP and PCR, as well as the goal to understand mechanisms that link individual and team-level effects, we approached our research questions with a critical realist stance. The study draws on Nonaka (1994)'s theory of organizational knowledge creation to guide data collection and analysis as well as our theorizing efforts.

Our results contribute to extant literature of two research areas. First, research on the effectiveness of ISD techniques gains insights from our study in that we show that PP and PCR facilitate different knowledge conversion mechanisms, both on an individual and on a team level. We show that they may substitute each other with regard to the four-eyes principle on a dyad level, but may act as complements regarding knowledge creation and diffusion on a team level. They collectively enable socialization, externalization, combination, and internalization (Nonaka, 1994). Research assessing the effectiveness of ISD techniques must therefore account for such differences that go beyond impacting code quality. Second, we contribute to research that investigates how ISD teams come to benefit from the knowledge of their single members (e.g., Maruping et al., 2009; Nemanich et al., 2010; Spohrer et al., 2012) by showing that the use of different development techniques impacts a team's ability to create knowledge from the contributions of its single members.

The next section provides a brief review of existing literature on PP and PCR and introduces Nonaka (1994)'s theory of organizational knowledge creation in the context of ISD teams. Subsequently, the study's research design is presented. Then, the findings regarding individual and team-level effects of PP and PCR on knowledge creation are presented and discussed.

2 Foundations

Pair programming and PCR represent two collaborative software development techniques that share the goal of ensuring high quality through the adherence to the four-eyes principle. While PP has drawn recent attention in the fields of Information Systems and Software Engineering (cf. Dingsøyr et al., 2012; Balijepally et al., 2009) as well as in studies focusing on the education of software engineers (cf. Salleh et al., 2011), little is known about the application of PCR in for-profit settings.

Pair programming refers to a software development technique with two developers simultaneously working in front of a single computer (Balijepally et al., 2009). There are two roles which are frequently switched. One developer is in the role of the *driver*, actively developing the solution, whereas the so-called *observer* is "observing the work of the driver and identifying tactical and strategic deficiencies in their work" (Williams, 2000, p. 3). Existing work on PP is primarily concerned with pairing developers' work effectiveness and efficiency, mostly compared to solo programming as the point of reference (Salleh et al., 2011; Dybå & Dingsøyr, 2008; Dingsøyr et al., 2012).

Peer code review is a second development method that relies on the four-eyes principle. After a piece of software code is developed or modified by a single software engineer, it is submitted to a code review system through which peer developers are invited to review and propose changes, as well as highlight defects, inaccuracies, and potential improvements. Based on this code-specific feedback, the original author then revises the code. These submit-and-feedback cycles can be repeated several times until the involved peers are eventually satisfied. Only then, the code is committed to the common code line. All communicational interactions are documented in this system and accessible by other participants. Frequent, iterative peer code reviews of small pieces of code have been successfully used as a quality assurance technique in open source communities (Rigby & Storey, 2011; Liang & Mizuno, 2011). For-profit organizations, by contrast, have often relied on occasional, formal code inspections of completed artifacts that show very different characteristics (Rigby et al., 2012). Scholars have actually argued that PCR shares more commonalities with PP than with traditional code inspections (Rigby et al., 2012).

With a few exceptions in the context of open source communities (Rigby & Storey, 2011; Liang & Mizuno, 2011), only limited research exists that studies PCR. Moreover, given that both techniques implement the four-eyes principle, surprisingly few attempts have been made to contrast PP and PCR (Müller, 2004, 2005; Rigby et al., 2012; Schmidt et al., 2012) . Generally, research on both, PP and PCR has mostly focused on direct influences on code quality and programming efficiency. However, PP and PCR may change the way in which software developers work together and exchange information. Thus, these techniques are likely to alter knowledge flows and learning processes within ISD teams (Vidgen & Wang, 2009; Rigby et al., 2012).

While other disciplines have established knowledge on how teams learn (cf. Edmondson et al., 2007; Wilson et al., 2007; Goodman & Dabbish, 2011), only little is known about how ISD teams actually come to benefit from the knowledge of their single members (Sarker et al., 2011; Spohrer et al., 2012). Our study adds to the latter stream of research by examining how using PP and PCR affects knowledge creation on both a dyad and a team level.

We base our research on organizational knowledge creation theory (Nonaka, 1994; Nonaka et al., 2006; Nonaka & Von Krogh, 2009) for two reasons: first, by its very nature it is concerned with activities of individuals that create knowledge through theoretically established mechanisms; second, it addresses the distribution and enhancement of knowledge across levels within organizations.

In particular, Nonaka (1994) suggests that knowledge resides on a continuum between being explicit and tacit. Explicit knowledge can be easily articulated, codified, and communicated to others, whereas tacit knowledge is implicit and embodied (Nonaka & Von Krogh, 2009). Organizational knowledge creation theory essentially posits that new knowledge is created by individuals who convert and restructure existing knowledge. On the one hand, internalization and externalization mech-

anisms are central to converting explicit to tacit knowledge and vice versa. On the other hand, socialization and combination refer to the restructuration of tacit and explicit knowledge respectively (Nonaka & Von Krogh, 2009). Organizational knowledge creation theory further suggests that knowledge is created and distributed in organizations through a sequential process of socialization, externalization, combination, and internalization that involves a continuously increasing number of actors (Nonaka, 1994).

3 Research Design

The goal of this study is to explain how both PP and PCR influence knowledge creation in ISD teams on a dyad as well as a team level.We approach this question from a philosophical perspective of critical realism (cf. Archer et al., 1998; Mingers, 2004). In doing so, we acknowledge that there are real structures and generative mechanisms underlying the observed socio-technical phenomena, but that we can neither be sure to observe all existing phenomena, nor that the observed events are actually all the possible consequences of these mechanisms. Our approach, thus, does not aim at testing a-priori articulated hypotheses in a positivist manner, nor does it try to understand a single case only in an interpretive manner. Instead, we explore in a context-sensitive way "what the world must be like" in order for the observed phenomena to occur (Mingers, 2004, p. 92).

In order to understand the underlying mechanisms of knowledge creation in the context of PP and PCR, a critical realist lens requires an examination of the structures as well as the functions of the involved entities (Mingers, 2004). Following accepted guidelines for critical realist case studies (Wynn & Williams, 2012), we examined ISD teams of different sizes and task contexts that all rely on PP, PCR, or both in their daily ISD tasks. Table 1 provides an overview of our cases with pseudonyms for the teams, their size and qualitative values for how frequently they apply each technique (PP / PCR Use) as well as rough descriptions of the software products they develop. The teams all belong to one large software vendor that has recently undergone organizational changes which make it particularly interesting for our study: the majority of its ISD teams are now working based on Scrum and management propagates the use of agile ISD techniques where possible. As such, all of the studied teams were familiar with peer-based quality techniques and several of them were trained in agile techniques during winter and spring 2011/2012. Moreover, the teams were free in their decisions if and how to adopt the different practices. The investigation of teams that developed a variety of different products within a single software vendor allowed us to examine different ways and results of applying PP and PCR while the context of the teams remained similar regarding their organizational background.

Data was collected from nine ISD teams through 13 semistructured face-to-face interviews with key informants of typically one hour. The data collection took place during summer 2012. In general, key informants are assumed to make more detailed statements than their colleagues about aspects of social structures that concern their roles within these structures (Houston & Sudman, 1975). Thus, we selected informants based on their roles and long history in the ISD teams: Scrum masters, agile method trainers, and senior developers were appropriate key informants as they actively developed code in the teams, but were also sensitized to the use of agile techniques. These key informants provided extensive details about the teams and how PP and PCR were actually applied there (cf. Table 1, Interviews). Informants were asked to describe single occurrences of applying the techniques themselves as well as the use within the entire teams. In particular, we elicited information about structure, context, and events of the phenomena of interest (Wynn & Williams, 2012). In addition, two of us joined central trainings for PP and PCR at the case site and obtained the relevant course materials so that we could also understand how these techniques were taught and defined in the company.

Interviews were audiotaped, transcribed, and analyzed in NVivo already during the process of data

Aspect/Team	INSTALL3	UI	CORRECT	BPM	INSTALL1
PP Use	high	high	high	high	high
-- Rotation	-- high	-- low	-- high	-- high	-- low
PCR Use	no	low	moderate	high	high
-- Reviewers	N/A	-- single	-- single	-- multiple	-- multiple
-- Readers	N/A	-- no	-- no	-- multiple	-- multiple
Product	Version Verification for Products	User Interface for Software	Code Correction Maintenance	Process Modeling Software	Installation Framework
Team Size	3	10	5	10	10
Interviews	SD	SM/TR	SD	SM/TR, SD	SD,SD

Aspect/Team	INSTALL2	PORTAL	REVIEW	COMPILE	
PP Use	moderate	low	low	no	
-- Rotation	-- moderate	-- moderate	-- low	N/A	
PCR Use	moderate	high	high	high	
-- Reviewers	-- single	-- multiple	-- multiple	-- multiple	
-- Readers	-- no	-- multiple	-- multiple	-- no	
Product	Client Installation Suite	Web Portal	PCR Tool	Custom Language Compiler	
Team Size	4	4	8	8	
Interviews	SD	SD	SM, SD, SD	SM	

SM: Scrum Master; TR: Agile Technique Trainer; SD: Senior Developer; /: Informant with multiple roles

Table 1. Overview of Cases

collection. We went back and forth between literature and empirical data (Eisenhardt, 1989) and used our theoretical lens to focus on described events that related to the knowledge conversion processes defined by Nonaka (1994). Constantly comparing theory to data, we retroduced the described events and structures to causal links with the underlying characteristics of the two techniques and their use (Wynn & Williams, 2012). With such concepts in mind, we went into the next interviews to corrobate and refine our explanations. We were granted access to documents about the teams' characteristics, products, and task areas and had several informal conversations with experienced managers and agile methods trainers, thus being enabled to triangulate the findings from the semi-structured interviews.

4 Findings

Subsequently, we elaborate on the different use patterns of PP and PCR on a dyadic and on a team level. We explain their meaning for knowledge creation in these teams and analyze the resulting findings with regard to the relationship between PP and PCR. In addition, we illustrate the insights gained from our extensive data analysis through selected quotes from our interview partners. Figure 1 depicts the results of our analysis. We first turn to the analysis of effects on a dyad level (Section 4.1, Figure 1 bottom left & center), then to effects on a team level (Section 4.2, Figure 1 top left & center), to finally turn to the analysis of effects of combined use of both techniques (Section 4.3, Figure 1 bottom & top right).

4.1 Knowledge Creation Processes between Individuals

All teams that used PP as a technique for their development tasks agreed on its usefulness for complex tasks that required knowledge of more than a single developer. Emphasis was placed not only on the actual code development, but also on analyzing the problem and conceptualizing a solution together. Learning from others was seen as a central reason for using PP. Participants primarily reported two learning effects during PP sessions: first, understanding others' viewpoints and knowledge on a certain problem as they exchanged opinions, solutions, and code; and second, observing their peers in the driver role to learn from their actions, not only from their statements.

"Working on a topic together, I think there is a lot of knowledge transfer. I do realize how my colleagues work, especially in Eclipse. [...] You really learn something!" (SD INSTALL1)

From a knowledge creation perspective, these activities correspond to two knowledge conversion mechanisms: socialization and externalization. On the one hand, developers express their opinions on what their peer is doing or saying. In doing so, the developers make tacit knowledge they hold in the particular domain explicit and relate it to the peer's actions or statements. However, especially negative opinions about a peer's work are not necessarily made explicit. Instead, peers rather comment and change the way the task is approached. Thereby, they act based on their implicit knowledge rather than expressing it verbally. That is, pairs convert tacit knowledge from both individuals to further tacit knowledge required for task completion and thus socialize it on the level of a particular pair (Figure 1 bottom left).

When teams use PCR, our findings revealed that authors usually submit only small code changes for better overview and lower complexity for the reviewer. As a reviewer, they evaluate source code as soon as possible and especially with regard to code conventions of their team, such as naming and structuring conventions to ensure readability, testability, and maintainability. The author then reads the comments and critique, addresses them and implements them if deemed appropriate. The conversations that emerge from these evaluations take place in written form via the PCR system. Peer code review thereby forces both, reviewer and author, to externalize their knowledge and beliefs in order to give arguments for or against additional changes of the code (Figure 1 bottom center). Several informants acknowledged difficulties with giving feedback via PCR that goes beyond directly observable issues. As such, the need to convey all relevant aspects in written form was found to make it harder to recommend different solution strategies via PCR. Moreover, participants found it hard to determine alternative solutions because they were not involved in creating the original one.

"In code reviews, the focus is more on the code; clean code aspects. But to determine if it was the right way to go... You'd have to read a lot of code to understand that... You actually can't." (SM UI)

A comparison of the two techniques and their use on a dyadic level, consequently, reveals that both rely on verbally externalizing knowledge as a primary quality improvement stimulus. However, while the developers have to make the relevant knowledge explicit in order to give convincing arguments in PCR, they can rely on tacit knowledge shared through social cues and immediately transformed into actions during PP. Reviewers in PCR are not involved in the creation of the solution in the first place and the author cannot easily externalize this knowledge. Consequently, reviewers lack the tacit knowledge about the genesis of the code and can only examine the result. From a knowledge creation perspective, PCR relies on pure knowledge externalization on a dyadic level while PP also provides plenty of opportunity for socialization (cf. Figure 1 bottom left & center).

4.2 Knowledge Creation Processes on a Team Level

So far, we examined the discrete feedback and knowledge creation mechanisms of PP and PCR on a dyadic level. According to Nonaka (1994), organizational knowledge creation not only requires knowledge to be converted from and to different forms, but also that an increasing number of actors becomes involved. Therefore, we examined how both PP and PCR come to involve more than only two members of the ISD teams.

With regard to team-level use patterns of PP, teams not only differed in the frequency of their use but also in the rotation of members across pairs (cf. Table 1 PP Use & Rotation). In some teams, single members usually had the same pairing partners while pairs mixed up frequently in other teams. We found that such rotation is the primary mechanism for propagating knowledge via PP. On a dyadic level, PP primarily fosters the extension of pair members' tacit knowledge through socialization and verbal articulation of task-related knowledge. Moving to the next pairing partner, developers can

contribute their extended tacit knowledge as it is embodied in their actions and understanding. However, the verbally articulated and externalized knowledge from previous sessions is not documented. As a consequence, developers can only transfer those aspects of the explicit knowledge to the next pairing partner that they understand and remember. Details and nuances that could not be related to their existing knowledge are not transferred between pairs with different developers. Socialization is, consequently, the primary knowledge creation and conversion mechanism of PP on a team level (cf. Figure 1 top left).

"[...] in PP you discuss a lot and develop a common understanding over time. And in the end, both understand it, but it might be so complex that nobody else can understand it anymore. (SM REVIEW)

In all studied cases, access to the PCR system of the teams was unrestricted for all employees of the company. Consequently, team members could see all code changes an author had submitted to the system and had to decide if they were going to read or even review them. Thus, three exhaustive roles emerged for all team members involved in a code change: one author, one or more reviewers, and no or more passive readers. The use patterns of the single teams then determined how these actors engaged in knowledge creation and conversion. We discovered three distinct use patterns for PCR on a team level: (1) the author explicitly invites one reviewer and only these two actors examine the code; (2) multiple invited team members perform code reviews; (3) the author does not explicitly ask for reviews from a single team member but the peers select code changes in the PCR system on their own and decide to review them (cf. Table 1 PCR Reviewers & Readers). We find that associated knowledge creation processes differ across these patterns. On a team level, reviewers externalize their knowledge similar to dyadic interactions. However, in settings (2) and (3), the author is required to take explicit knowledge into account from multiple reviewers. Thereby, the author combines the expertise and explicit statements of reviewers to create a new and enhanced outcome:

"In the end, I set my focus on clean code. [...] Others watched for typos, my soft spot, others said formatting is important. [...] And of course, there are the functional things you are interested in." (SD PORTAL1)

"[If you add a comment in the PCR system,] you have to refine it so that it is understandable. The advantage is that it can still be understood in one year." (SM REVIEW)

Moreover, four of the teams we studied had multiple reviewers or additional readers involved in PCR (cf. Table 1). These readers and reviewers become aware of the feedback given by other team members. Over time, they get used to frequently remarked issues and mistakes and consequently adapt to this feedback in their own code in order to prevent future issues in PCR. In other words, the social presence of readers or additional reviewers fosters internalization of explicit feedback given by reviewers and thereby the development of commonly accepted norms.

"Compared to older code, you realize it gets increasingly similar. [...] And you say, ok, now that I've said 100 times that I'd throw a NullPointerException [...] colleagues start to do it and then they spread the word, and then a third one does it. That's cool!" (SD BPM1)

In summary, we find that PP primarily stimulates socialization and, less intensively, also externalization activities on a dyadic level. On a team level, this effect induces a focus on creating, spreading, and aligning tacit knowledge across the team through socialization. By contrast, PCR primarily fosters externalization on a dyadic level, but translates to a more differentiated effect that stimulates externalization, combination, and internalization on a team level (cf. Figure 1 top left & center). It is central to both these translations that further actors get involved through rotation in PP and additional reviewers as well as readers in PCR. Indeed, we find that the frequency of applying one of the techniques results in occasions for knowledge creation, but the distinctive use patterns with more or less people involved are equally important with regard to which knowledge conversion processes

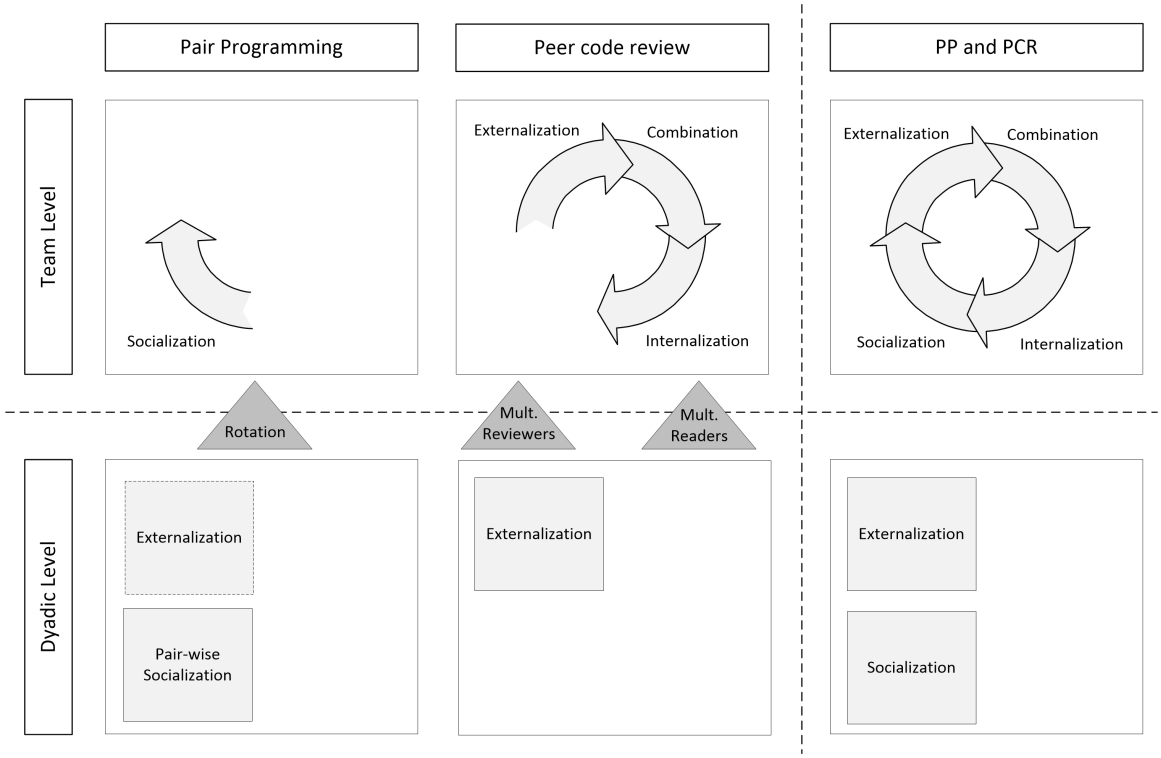

Figure 1. Impact of PP and PCR on Knowledge Creation in ISD Teams

take place on a team level. We elaborate more on this aspect in the following section and analyze the potential of isolated or combined use of PP and PCR.

4.3 Complementarity of Peer-Based Quality Techniques

Proponents of PCR have argued that PCR may act as a substitute for PP (Paulk, 2001; Müller, 2004; Rigby et al., 2012). Generally, ISD techniques may substitute each other if they are structurally and functionally equivalent by involving the same people (structural equivalence) and creating the same results (functional equivalence). Our prior analysis has shown that there are functional and structural differences between the use patterns of the techniques. PP may or may not involve rotating pairs of developers that engage in socialization and, on an individual level, in externalization of tacit knowledge. PCR, in contrast, can encompass the entire team as reviewers or readers for a single code change. With more members involved, it stimulates externalization of tacit as well as combination and internalization of explicit knowledge. As a consequence, substitutive effects should be expected only with regard to the externalization of tacit knowledge (functional equivalence) when similar numbers of peers are involved (structural equivalence). This is in line with our empirical data: equivalent effects are mainly reported with regard to error detection. Error detection results from giving feedback on the source code and forces the reviewer to make the context explicit. In fact, we find that teams who typically rely on a single, invited reviewer in PCR (cf. Table 1 Reviewers) saw no need to use PCR if the code had already been developed through PP. By contrast, teams that rely on several, self-assigned reviewers, and even passive readers in PCR stated that there are merits beyond error prevention that legitimate the additional use of PCR.

"[...] if a team uses pair programming, there is nothing against this change being reviewed after that. Pair programming is done with two people, but maybe a third one has another opinion." (SD REVIEW)

"And after pairing, [...] you could submit it. But my colleagues and I usually put it to review to bring everyone up to date. If one is interested, he reads it [...] and can still give feedback." (SD BPM)

Organizational knowledge creation theory (Nonaka, 1994) posits that for persistent knowledge creation to occur, the processes of socialization, externalization, combination, and internalization must take place in sequence and with an increasing number of actors involved. Against this backdrop, our findings suggest that PP and PCR may complement each other. On the one hand, PP was found to stimulate socialization and verbal articulation of tacit knowledge in the group and, thereby, helps to bring developers with different backgrounds to a commonly accepted basis of understanding. On the other hand, PCR stimulates written and documented articulation of tacit knowledge of several reviewing team members and fosters knowledge combination by the author. Finally, the visibility of feedback to additional readers in PCR allows for internalizing it through the development of team-wide norms. For persistent knowledge creation to occur on a team level, ISD teams can consequently benefit from applying both techniques (cf. Figure 1 top right).

5 Discussion

This study set out to better understand the effects of PP and PCR on knowledge creation in ISD teams. The results show that on a dyadic level (i.e., on the level of two individuals working together) and if applied separately, PP fosters socialization, whereas both techniques support different types of externalization. In particular, PP triggers verbal communication that may externalize some of the developers' implicitly held knowledge, while PCR requires the codification of knowledge. Thus, combining PP and PCR on a dyadic level may help software developers create new knowledge through close collaboration, as well as eliciting implicit knowledge and making it available for future reference. However, our results suggest that, if applied on a dyadic level only, PP and PCR fail to enable the combination of explicit knowledge and its internalization.

Interestingly, our findings reveal that if PP and PCR are elevated to the team level through pair rotation (in the case of PP) or multiple reviewers and readers (when applying PCR), additional knowledge creation effects can occur. As such, PP with frequently rotating pairs lowers the opportunity for externalization because most of the verbally expressed knowledge cannot be transfered to new contexts if it is not codified or internalized. By contrast, multiple reviewers that externalize knowledge through a PCR system enable the combination of different sources of explicit knowledge, whereas internalization on a team level can be achieved if reviews are read by more than one team member. Thus, if PP and PCR are combined and if mechanisms are in place that ensure the participation of multiple team members in the process (i.e., pair rotation, multiple reviewers, multiple readers), then the two techniques may collectively induce all four knowledge conversion processes. Following Nonaka (1994), this repeated cycle of converting and reproducing tacit and explicit knowledge on an increasing scale fosters persistent knowledge creation on a team level. Thus, the results of our study show that while PP and PCR are substitutes on a dyadic level with regard to implementing the four-eyes principle, they may complement each other on a team level. In other words, by collectively enabling knowledge creation within ISD teams, the value derived from combining PP and PCR may be higher than the summative value of both techniques when applied separately.

By improving our understanding of the impact of PP and PCR on knowledge creation within ISD teams, this study makes several contributions to extant literature. Previous research on PP and PCR has mostly focused on direct implications on code quality that result from the implementation of the four-eyes principle (Schmidt et al., 2012). By contrast, our study is among the first to comprehensively examine the implications of PP and PCR on team-level knowledge creation. This is important because differential team-level benefits of PP and PCR respectively force us to question the widely assumed substitutive nature of the two techniques. Indeed, by drawing on the theory of

organizational knowledge creation, we were able to show that with regard to team-level knowledge creation, PP and PCR may not act as substitutes, but rather complement each other.

By studying the varying implications of PP and PCR, this study adds to first efforts to compare the two techniques (Müller, 2004, 2005; Rigby et al., 2012). While prior studies relied on student experiments to evaluate the effectiveness and efficiency of the two techniques in improving code quality, our work complements this research by providing a deeper understanding of how applying the two techniques affects ISD teams. Specifically, our findings suggest that both, direct and indirect implications of PP and PCR should necessarily be considered when examining the effects of PP and PCR.

Prior research on ISD teams found that knowledge may not be homogeneously distributed within these teams and called for a more comprehensive examination of the interplay of individual and group-level knowledge sharing processes (Sarker & Sarker, 2009). Recent research provided first insights on these processes by showing that PP, and particularly pair rotation, may enable knowledge sharing and team learning (Vidgen & Wang, 2009). Following Nonaka & Von Krogh's call for research on a work team level at the intersection of social practice and knowledge creation (Nonaka & Von Krogh, 2009, p. 647-648), our study contributes to these findings by simultaneously studying PP and PCR as two collaborative software development techniques, as well as the interplay of dyadic and team-level implications of both techniques.

Our findings provide important insights for ISD practitioners. ISD teams and decision makers within ISD organizations learn that it may not be enough to evaluate the benefits of PP versus PCR only by comparing direct and short-term quality gains with the costs resulting from increased human-capital-intensity. In particular, while PP and PCR substitute each other with regard to implementing the four-eye-principle, practitioners should be aware that such a short-term cost-benefit analysis may fall short in considering the indirect team-level effects of collaborative software development techniques. More specifically, responsibles in ISD organizations learn that in order to increase team-wide knowledge creation, it may be necessary to combine techniques that foster socialization among developers with others that ensure codification of knowledge as well as the possiblity to combine and internalize knowledge from different sources. Our findings suggest that in combination, PP with rotating pairs and PCR with multiple reviewers and readers may indeed foster knowledge creation within ISD teams.

Several limitations have to be kept in mind when interpreting the findings of our study. First, we examined effects on knowledge creation on a team level associated with the application of PP and PCR. While such a knowledge creation perspective on social practices within ISD teams provided important insights, future research may study PP and PCR within ISD teams from different angles. For instance, previous research proposed that the feedback processes inherent in these techniques may help ISD teams establish and enhance their transactive memory (Schmidt et al., 2012). Combining these two approaches to study collaborative software development techniques may yield further insights on their team-level implications. Moreover, Nonaka (1994)'s knowledge conversion mechanisms have so far been mostly applied to scenarios of knowledge creation on an organizational level, whereas we draw on this theory to understand team-level effects. While the appropriate level of analysis of Nonaka (1994)'s theory of knowledge creation has been discussed controversially in literature (Gourlay, 2006; Griffith & Sawyer, 2010), it helped us understand and explain knowledge creation in the context of ISD teams. In addition, it has to be kept in mind that our findings are largely based on interviews with key informants. While this approach was well suited for our study because it enabled us to examine a larger number of teams that apply PP and PCR with varying intensity, future research may gain deeper insights and a more comprehensive understanding by conducting in-depth case study research within a smaller number of ISD teams that represent the different use patterns.

6 Conclusion

While pair programming and peer code review may be treated as partial substitutes on a dyadic level, we showed that they may complement each other regarding knowledge creation on a team level. Specifically, our study shows that the complementarity of the two techniques on a team-level knowledge creation stem from their potential to collectively enable socialization, externalization, combination, and internalization processes. Consequently, ISD teams that decide to use only one of these techniques based on the sole comparison of dyad-level effects may therefore miss important chances for knowledge creation.

References

Archer, M., Bhaskar, R., Collier, A., Lawson, T., & Norrie, A., Eds. (1998). *Critical realism: Essential readings*. Routledge, London.

Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are two Heads better than one for Software Development? The Productivity Paradox of Pair Programming. *MIS Quarterly*, 33(1), 91-118.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *The Journal of Systems & Software*, 85, 1213-1221.

Dybå, T. & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10), 833 - 859.

Edmondson, A. C., Dillon, J. R., & Roloff, K. S. (2007). Three Perspectives on TeamLearning. *The Academy of Management Annals*, 6, 269-314.

Eisenhardt, K. M. (1989). Building Theories From Case Study Research. *The Academy of Management Review*, 14(4), 532-550.

Goodman, P. S. & Dabbish, L. A. (2011). Methodological Issues in Measuring Group Learning. *Small Group Research*, 42(4), 379-404.

Gourlay, S. (2006). Conceptualizing Knowledge Creation: A Critique of Nonaka's Theory. *Journal of Management Studies*, 43(7), 1415-1436.

Griffith, T. L. & Sawyer, J. E. (2010). Multilevel knowledge and team performance. *Journal of Organizational Behavior*, 31(7), 1003-1031.

Houston, M. J. & Sudman, S. (1975). A methodological assessment of the use of key informants. *Social Science Research*, 4(2), 151 - 164.

Liang, J. & Mizuno, O. (2011). Analyzing Involvements of Reviewers Through Mining A Code Review Repository. In *Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement* Nara, Japan.

Maruping, L. M., Zhang, X., & Venkatesh, V. (2009). Role of collective ownership and coding standards in coordinating expertise in software project teams. *European Journal of Information Systems*, 18(4), 355-371.

Mingers, J. (2004). Realizing Information Systems: Critical Realism as an Underpinning Philosophy for Information Systems. *Information and Organization*, 14(2), 87-103.

Müller, M. M. (2004). Are Reviews an Alternative to Pair Programming? *Empirical Software Engineering*, 9(4), 335-351.

Müller, M. M. (2005). Two controlled experiments concerning the comparison of pair programming to peer review. *Journal of Systems and Software*, 78, 166-179.

Nemanich, L., Keller, R., Vera, D., & Chin, W. (2010). Absorptive Capacity in R & D Project

Teams: A Conceptualization and Empirical Test. *Engineering Management, IEEE Transactions on*, 57(4), 674 -688.

Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1), 14-37.

Nonaka, I. & Von Krogh, G. (2009). Tacit knowledge and knowledge conversion: Controversy and advancement in organizational knowledge creation theory. *Organization Science*, 20(3), 635-652.

Nonaka, I., von Krogh, G., & Voelpel, S. (2006). Organizational knowledge creation theory: Evolutionary paths and future advances. *Organization Studies*, 27(8), 1179-1208.

Paulk, M. C. (2001). Extreme Programming from a CMM Perspective. *IEEE Software*, 18(6), 19-26.

Rigby, P., Cleary, B., Painchaud, F., Storey, M.-A., & German, D. (2012). Contemporary Peer Review in Action: Lessons from Open Source Development. *IEEE Software*, 29(6), 56-61.

Rigby, P. C. & Storey, M.-A. (2011). Understanding Broadcast Based Peer Review on Open Source Software Projects. In *Proceedings of the International Conference on Software Engineering* Honolulu, Hawaii, USA.

Salleh, N., Mendes, E., & Grundy, J. C. (2011). Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, 37(4), 509-525.

Sarker, S. & Sarker, S. (2009). Exploring Agility in Distributed Information Systems Development Teams: An Interpretive Study in an Offshoring Context. *Information Systems Research*, 20(3), 440-461.

Sarker, S., Sarker, S., Kirkeby, S., & Chakraborty, S. (2011). Path to ``Stardom'' in Globally Distributed Hybrid Teams: An Examination of a Knowledge-Centered Perspective using Social Network Analysis. *Decision Sciences*, 42(2), 339-370.

Schmidt, C. T., Spohrer, K., Kude, T., & Heinzl, A. (2012). The Impact of Peer-Based Software Reviews on Team Performance: The Role of Feedback and Transactive Memory Systems. In *Proceedings of the International Conference on Information Systems* Orlando, Florida, USA.

Spohrer, K., Gholami, B., & Heinzl, A. (2012). Team Learning in Information Systems Development - A Literature Review. In *Proceedings of the European Conference on Information Systems 2012* (paper 223). Barcelona, Spain.

VersionOne, I. (2011). State of Agile Survey 2011.

Vidgen, R. & Wang, X. (2009). Coevolving Systems and the Organization of Agile Software Development. *Information Systems Research*, 20(3), 355-376.

Williams, L. A. (2000). *The Collaborative Software Process*. PhD thesis, Department of Computer Science, University of Utah.

Williams, L. A. & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communincations of the ACM*, 43(5), 108-114.

Wilson, J. M., Goodman, P. S., & Cronin, M. A. (2007). Group learning. *Academy of Management Review*, 32(4), 1041-1059.

Wynn, Jr., D. & Williams, C. K. (2012). Principles for Conducting Critical Realist Case Study Research in Information Systems. *MIS Quarterly*, 36(3), 787 - 810.