Perceived Barriers to Effective Knowledge Sharing in Agile Software Teams

“You never really understand a person until you consider things from his point of view. Until you climb inside of his skin and walk around in it.” ~ Harper Lee, To Kill a Mockingbird

Shahla Ghobadi* & Lars Mathiassen†

*University of New South Wales, PO Box H58, Australia Square, Sydney 2052, Australia, email: s.ghobadi@unsw.edu.au, and †Georgia State University, PO Box 4015, Atlanta, GA 303302, USA, email: lmathiassen@ceprin.org

Abstract

While the literature offers several frameworks that explain barriers to knowledge sharing within software development teams, little is known about differences in how team members perceive these barriers. Based on an in-depth multi-case study of four software projects, we investigate how project managers, developers, testers, and user representatives think about barriers to effective knowledge sharing in agile development. Adapting comparative causal modeling (CCM), we constructed causal maps for each of the four roles and identified overlap and divergence in map constructs and causal linkages. The results indicate that despite certain similarities, the four roles differ in how they perceive and emphasize knowledge sharing barriers. The project managers put primary emphasis on project setting barriers, while the primary concern of developers, testers, and user representatives were project communication, project organization, and team capabilities barriers respectively. Integrating the four causal maps and the agile literature, we propose a conceptual framework with seven types of knowledge sharing barriers and 37 specific barriers. We argue that to bridge communication gaps and create shared understanding in software teams, it is critical to take the revealed concerns of different roles into account. We conclude by discussing our findings in relation to knowledge sharing in agile teams and software teams more generally.

Keywords: Knowledge sharing, knowledge sharing barriers, agile development, agile teams, software development, comparative causal capping, cognitive modeling, qualitative study
Introduction

Software development is a collaborative process where success depends on effective knowledge sharing (Andres, 2002, Curtis, et al., 1988, Hansen & Kautz, 2004, Joshi, et al., 2007, Sawyer, et al., 2008, Walz, et al., 1993, Williams, 2010). This is particularly true in agile development where great emphasis is placed on communication involving diverse stakeholders through practices such as joint-application design sessions and customer focus groups (Chan & Thong, 2009, Chau, et al., 2003, Cockburn, 2006, Conboy, et al., 2010, Corvera Charaf, et al., 2012, Kautz, et al., 2007). Not surprisingly, to support agile practices in software development, the Agile Manifesto lists twelve principles (http://agilemanifesto.org/principles.html), such as ‘focus on the customer’, ‘collaborate regularly’, ‘communicate face-to-face within the team’, and ‘have regular team introspection’, that inherently require effective knowledge sharing practices between team members.

Nonetheless, knowledge sharing in agile teams is challenging (Chan & Thong, 2009, Cockburn, 2006, Conboy, et al., 2010, Nerur, et al., 2005, Ramesh, et al., 2010). For example, it can be challenging to increase users’ motivation to share specific business knowledge with developers (Conboy, et al., 2010), to manage cross-functionality and the variety of social identities involved in software development (Ghobadi, 2011, Highsmith, 2002), and to encourage and facilitate sharing tacit knowledge amongst team members (Bjørnson & Dingsøyr, 2008). In addition, dynamic business environments have seen many software organizations adopt approaches that blend agile principles with distributed development concepts to reap the benefits of both (Ramesh, et al., 2006, Ramesh, et al., 2012). This has, however, increased the knowledge sharing challenges agile teams face (Ramesh, et al., 2006).

While efforts have been undertaken to study barriers to knowledge sharing in software teams in general and agile teams in particular (Chau & Maurer, 2004, Chau, et al., 2003, Levy & Hazan, 2009), these barriers may not be addressed without taking into account the concerns of the involved stakeholders (Mähring, et al., 2008, Newman & Robey, 1992, Robey, et al., 2001). In fact, due to diverse role expectations and communication gaps, software team members may have perceptual differences on issues such as project success, project failure, risk factors, and appropriateness of development methodologies (Huisman & Iivari, 2006, Keil, et al., 2002, Levesque, et al., 2001, Linberg, 1999). Similarly, different roles such as managers, developers, testers, and user representatives may have perceptual differences regarding barriers to effective knowledge sharing practices. For example, users may perceive developers’ lack of knowledge about their field a barrier to knowledge sharing. Developers may, in turn, perceive political issues and inconsistencies in users’ description of functional requirements to be major barriers.
As a result, developers may defer close interaction with users until they have negotiated an agreement about overall project goals and major requirements, and users may, in turn, link developers’ lack of knowledge to their busy schedules and hold back on interacting with them unless developers take the initiative. Perceptual differences and expectations may, in particular, be present in teams where extreme dedication in adhering to or interpreting agile values and principles reinforce communication-related difficulties (Persson, et al., 2012) and inconsistencies (Boehm, 2002, Pikkarainen, et al., 2008, Turk, et al., 2005). For example, user representatives may expect developers to be constantly open to new system requirements (referring to the ‘welcome change’ agile principle), and thus they may interpret developers’ inflexibility in welcoming change as a barrier to effective knowledge sharing practices. However, developers may disagree, arguing that besides the importance of having certain levels of control and accountability in projects, welcoming change requirements late during development may force them to work extra hours, contradicting the agile principle that values individuals more than processes (Boehm, 2002). Therefore, developers may consider user representatives’ lack of attention to specifying requirements as early as possible a key barrier to effective knowledge sharing.

We argue these perceptual differences must be taken into account to create shared understanding among team members as a key to performance in organizations (Laukkanen, 1994). Different perceptions may result in divergent actions to overcome barriers and in unsatisfactory outcomes that eventually may give rise to conflicts between stakeholders. Obviously, such behaviors establish unproductive communication practices and unhealthy relationships across stakeholder groups and they may result in project delays and less than optimal software solutions. Therefore, stakeholders must explore differences in perception and come to an agreement on barriers to effective knowledge sharing practices. In fact, the process of exploring and appreciating such differences is a necessary first step for software teams to mitigate the knowledge sharing barriers they face.

In spite of calls for a broadened view of barriers to knowledge sharing within software teams in general (Levina, 2005) and agile teams in particular (Chan & Thong, 2009), perceptual differences among different roles remain relatively unexplored. In fact, a systematic investigation of perceptual differences of software team members regarding barriers to knowledge sharing has yet to receive significant attention. Against this backdrop, we utilized a qualitative methodology, Comparative Causal Mapping (CCM) (Laukkanen, 1994) to analyze data from four agile teams within two software companies. Distinguishing between the roles of ‘project manager’, ‘developer’, ‘tester’, and ‘user representative’, we constructed role specific causal maps on knowledge sharing barriers in agile teams. The results indicate that despite certain similarities across mappings, the four roles differ in how they perceive and emphasize
barriers. Integrating the causal maps with the agile literature, we also propose a conceptual framework of barriers to effective knowledge sharing in agile teams. These insights provide valuable contributions to current theory on knowledge sharing in agile teams and software teams more generally.

**Theoretical Background**

The extant literature demonstrates several attempts to understand knowledge sharing barriers or drivers within software teams (Ghobadi & D'Ambra, 2013, Joshi, et al., 2007, Kotlarsky & Oshri, 2005, Oshri, et al., 2008, Pee, et al., 2010). Focusing on software teams in general, Joshi et al. (2007) employed a connectionist epistemological perspective to reveal the crucial role of source's credibility and extent of communication in shaping knowledge transfer within teams. In another study, Ghobadi and D’Ambra (2013) described the dynamics through which outcomes (goals and rewards), means (task, resource, and role), and boundary (sense of identity, friendship, proximity) interdependencies drive simultaneously cooperative and competitive behaviors that in turn influence high-quality knowledge sharing in cross-functional software teams.

Turning to agile teams, extant literature suggests several process and context-related factors drive knowledge sharing (Augustine, et al., 2005, Cockburn & Highsmith, 2001, Conboy, et al., 2010, Conboy & Morgan, 2011, McAvoy, et al., 2012, Ramesh, et al., 2010, Vidgen & Wang, 2009) such as lightweight postmortem styles (Augustine, et al., 2005, Dingsøyr & Hanssen, 2003) and sufficient documentation (Conboy & Morgan, 2011, Karlsen, et al., 2011, Paetsch, et al., 2003). The agile literature also acknowledges the driving impact of motivation-related factors (e.g., career consequences, top management support), collaborative factors (e.g., encouraging agile coaching, training, practices such as pair programming and collective code ownership) (Chan & Thong, 2009, Chau & Maurer, 2004, Chau, et al., 2003, Conboy, et al., 2010, Melnik & Maurer, 2004), and ability-related factors (e.g., self-efficacy and experience culture) (Chan & Thong, 2009).

While the general software development literature provides us with more solid theoretical foundations in this area (as discussed above), the agile literature is still fragmented with limited analyses based on and contributing to rigorous theoretical perspectives (Dingsøyr, et al., 2012). This void of research about knowledge sharing in agile development is surprising given the high emphasis on communication and collaboration. Further, prior studies have paid little attention to specific team aspects, such as perceptual differences regarding knowledge sharing barriers in agile teams, thus hampering the development of a rigorous understanding of agile team characteristics. With this background, we designed our research based upon existing explanations of knowledge sharing within software teams in general and agile teams.

in particular to explore perceptual differences to knowledge sharing barriers across key roles in agile teams. We drew on organizational role theory (Katz & Kahn, 1978) with its roots in sociology (Biddle, 1986, Biddle, 1979) and offering insights for understanding behaviors and attitudes of individuals at work (Mähring, et al., 2008, Peppard & Ward, 2004). The theory asserts that behavior in a particular work role, referring to a distinct set of activities, is the result of organizational demands, social demands and personal demands on that role. Accordingly, an organizational role, indicated by a position title and specified by a job description, encompasses expected beliefs and behaviors attached to it. The focus of this theory on linking roles to beliefs and behaviors is, therefore, aligned with our interest in understanding differences between roles in perceiving knowledge sharing barriers in agile development.

We focused on four key roles in software development projects to understand the variety of perspectives and behaviors. Specifically, building on previous research on key roles and their interactions (Barki & Hartwick, 2001, Newman & Robey, 1992, Robey, et al., 2001), we targeted the four roles of ‘project manager’, ‘developer’, ‘tester’ and ‘user representative’, covering the recurrent actions performed during software development.

To sum up, informed by organizational role theory and the literature on knowledge sharing within software teams in general and agile teams in particular our research objectives are: (i) to explore the similarities and differences in how four key roles—‘project manager’, ‘developer’, ‘tester’, and ‘user representative’—perceive barriers to knowledge sharing in agile teams, and (ii) to integrate the findings into a conceptual framework that encompasses these perceptions and provides implications for managing agile development. To address these objectives, we investigate the following research question: How do key stakeholders in agile software development perceive barriers to effective knowledge sharing?

Research Method

Comparative Causal Mapping

CCM is a variant of cognitive mapping where respondents explain their causal assertions about a phenomenon through interview sessions (Laukkanen, 1994). It demonstrates the patterns of concepts and causal beliefs that are embedded in explicit statements of different groups. CCM is primarily intended and appropriate for comparative analyses between different types of organizational actors and pinpointing cognitive similarities or differences across those actors (Laukkanen, 1994, Laukkanen, 1998). Moreover, it best serves explorative and evocative interests with a focus on small or medium sample qualitative studies (Laukkanen & Päivi, 2013). CCM has been used to investigate perceptual similarities and
differences across individuals or groups (Budhwar & Sparrow, 2002, Chandra & Loosemore, 2010, Jenkins & Johnson, 1997, Tyler & Gnyawali, 2009) and it is therefore a suitable method for this research.

Adapting CCM to examine similarities and differences in perceiving knowledge sharing barriers across different roles, we considered the proposed stages for CCM (Chandra & Loosemore, 2010, Laukkanen, 1994, Tyler & Gnyawali, 2009): (i) predata and data collection stages (participant selection, interview design, symmetric data collection, follow-up interview sessions), (ii) analysis and post-data collection stage (establishing coding guidelines and higher order categories, identifying causal relationships, drawing maps’ based on reflected themes, analysis of maps, seeking feedback from experts). Our adaptation of each of these stages is elaborated below.

**Data Collection**

We applied theoretical replication logic to add confidence and robustness to our findings (Eisenhardt & Graebner, 2007, Yin, 2009). More specifically, we employed (i) replication strategy (Yin, 2009) to include a diverse range of perspectives on knowledge sharing barriers: by studying two companies and selecting participants to represent different roles, and, (ii) polar sampling (Eisenhardt & Graebner, 2007) to secure variance across selected projects and add insights related to project outcomes (Sonnentag, 1995, Yang & Tang, 2004): by selecting two completed projects within each company, one perceived to be high performing and one perceived to be low performing.

We selected two Australian software development companies (Alpha and Beta) that exemplify important characteristics for our study. The Australian software industry has shown increasing levels of market and technological growth (Griffith, 2011). Alpha and Beta are medium-size and they both follow collocated and distributed agile methodologies in their software projects. They have a long history of focusing on non-commercial software development with a recent profile of commercial projects, demonstrating their interest and established capability in expanding boundaries of their activities. We collected data through interview sessions with four key roles to discuss barriers to effective knowledge sharing in agile teams. We worked closely with the development manager of each company to identify and recruit eight interviewees (two project managers, two developers, two testers, two user representatives) from two of their recent agile projects. For example, the responsibility for individuals that held the role of project manager was to ensure customer expectations were addressed in the final product (both companies used the term ‘product owner’). For each company, we asked the development manager to select one high performing and one low performing project based on four indicators: (i) how stakeholders perceived
project performance, (ii) how the team dealt with challenges, (iii) how the project delivered expectations, and, (iv) how social relationships were experienced. The projects were completed within the last three months (at the time of the interviews), participation was voluntary, and participants were assured of confidentiality.

Respondents were asked about their perceptions regarding knowledge sharing practices during the considered agile project. The agile literature and existing frameworks on drivers of knowledge sharing within software teams guided the interviews, but we followed a multi-layered semi-structured interview guide (Appendix 1) to evoke new constructs and linkages, and, to create multiple occasions for access to human memory (Laukkanen, 1994). First, we collected background information about knowledge sharing during the specified project to explore key issues and jargon, and, to provide ‘anchor’ concepts for the next phase of the interview. For this, we asked open questions such as: ‘in what ways was knowledge sharing practiced?’ Second, based on responses, we asked probing questions to elicit further information about the concepts and relevant cause and effect relationships. Third, we did a final check by asking specific questions to point into key enablers and barriers. This check afforded interviewees a closing opportunity to remind us of any item that might have been missed or required further explanation. Interview sessions lasted from thirty minutes to one hour and were recorded and transcribed. Table 1 and 2 summarize project and respondent characteristics.

Table 1. Project Characteristics

<table>
<thead>
<tr>
<th>Description</th>
<th>Alpha</th>
<th>Beta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Performing Project (LA)</td>
<td>High Performing Project (HA)</td>
<td>Low Performing Project (LB)</td>
</tr>
<tr>
<td>Duration</td>
<td>6 months</td>
<td>4 months</td>
</tr>
<tr>
<td>Team Size</td>
<td>10 members</td>
<td>7 members</td>
</tr>
</tbody>
</table>

Table 2. Role Characteristics

<table>
<thead>
<tr>
<th>Role</th>
<th>Project Manager</th>
<th>Developer</th>
<th>Tester</th>
<th>User Representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>37.3 years</td>
<td>29.5 years</td>
<td>35.5 years</td>
<td>40.0 years</td>
</tr>
<tr>
<td>Education</td>
<td>Undergraduate: 75% Postgraduate: 25%</td>
<td>Undergraduate: 100%</td>
<td>Undergraduate: 100%</td>
<td>Undergraduate: 25% Postgraduate: 75%</td>
</tr>
</tbody>
</table>

After the analysis of the initial interviews, we returned to Alpha and Beta for follow-up interviews. The objective was to resolve ambiguities and validate interpretive accuracy and credibility of our interpretations (Guba & Lincoln, 1985). For example, there were quotes in which directionality was not clearly stated, rather the context of discussion had pointed to causality. We drafted specific graphical representations (causal maps) for each interviewee and allowed the individual to evaluate the concepts and linkages that were based on their own statements (all together 16 individual maps were drafted). We used semi-structured interviews to support this process by asking direct questions such as: “When you say dummy data for test was an issue do you mean it inhibited effective knowledge sharing?” This led to minor revisions in the individual causal maps.

**Data Analysis**

*Comparing Maps:* Despite the wide application of several variants of causal mapping, methodological discussions of CCM are relatively rare (Laukkanen & Päivi, 2013). We adapted Laukkanen (1994)’s approach for CCM using CMAP2 software by following four steps: (i) creating and using standard vocabularies, (ii) processing data for causal maps, (iii) constructing causal maps, and, (iv) analysis of causal maps. We leveraged different software packages such as Nvivo for coding and creating standard vocabularies, MS Excel for processing causal matrices and calculating indicators, and MS PowerPoint for creating visual causal maps.

In the first step, ‘creating and using standard vocabularies’, one researcher read all transcripts, grouped frequently mentioned words together, and inductively generated a list of preliminary codes. We represented each code by a label that summarized the meaning of a number of words or phrases used by the interviewees. We then examined the resulting coding scheme for theoretical or logical relevance against existing frameworks on drivers of knowledge sharing within software teams, without necessarily expecting the evoked maps to reflect these frameworks. Another researcher read the material to verify their face validity and assess the parsimony and coverage of the coding scheme. Once the coding scheme had stabilized, one researcher coded all transcripts and the other assisted by reading the material to verify the validity of the coding. Where disagreement occurred, discrepancies were resolved through discussion to minimize researcher biases. To estimate the reliability of the coding process, Scott’s pi (Scott, 1955)
was calculated using 8 of the 16 interview transcripts and 34 of the coding categories. A heuristic for content analysis is to require a reliability coefficient of approximately 0.75 or more (Holsti, 1969), and Scott’s pi for our study was estimated at an acceptable level of 0.86.

In the second step, ‘processing data for causal maps’, we examined all transcripts systematically to identify coded statements that imply an explicit cause-effect relationship by using keywords such as “so”, “if-then” and “because”. We recorded all causal statements and linkages in the actual language of the interviewees. We then developed a list of 60 perceived barriers (Appendix 2) by analyzing the identified causal statements and linkages and we categorized these barriers drawing on Leavitt’s organizational model of people, structure, technology and task (Leavitt, 1964) that has previously been applied to categorize software development risks (Lyytinen, et al., 1998, Persson, et al., 2009). As a result, we identified seven categories of barriers (see Appendix 2 for detailed information). For example, we categorized barriers related to geographical and time differences between team members (e.g., physical distance between team members’) as ‘team diversity’ barriers (discussion on the categories is provided in the Conceptual Framework section).

In the third step, ‘constructing causal maps’, we represented each identified ‘cause’ and ‘effect’ concept in the coded statements (from previous step) with the appropriate higher-order barrier constructs (from previous step). For example, the interview statement: “I think different cultures [cause] were challenging for knowledge sharing [effect]. We had agile culture that might be different with others, and universities are more bureaucratic and not agile,” was coded as ‘Agile Versus Non-Agile Cultures (client and development company)’ (categorized under project setting barriers in the second step). When drawing causal maps, we therefore established a link between ‘project setting’ and ‘effective knowledge sharing’ in the causal maps. In another example, when interviewees argued that unfamiliarity of team members (team diversity barriers) made them prefer working independently (tendency to work independently, categorized under team perceptions barriers), we established a link between ‘team diversity’ and ‘team perceptions’ barriers. Finally, we combined the cognitions of the four interviewees with similar roles and visualized them as a role specific causal map. This resulted in four causal maps that represented the perceptions of the four roles.

In the fourth step, ‘causal maps analysis’, we first analyzed the content of maps. Content analysis captures the subject and meaning of a communication that an individual (or group) perceives as being relevant. Our follow-up interviews provided member checks to validate interpretive accuracy and to check credibility and trustworthiness of each individual map (Guba & Lincoln, 1985). Next, we made sense of the maps and compared them (e.g., what are the key focus areas of project managers versus developers etc.) by analyzing their structure accompanied by the interview statements (Chandra &
Loosemore, 2010, Eden, et al., 1992). The objective of structural analysis is to examine the relationships among the items within a map and how they relate to each other (Chandra & Loosemore, 2010, Hodgkinson & Clarkson, 2005, Nadkarni & Narayanan, 2005). Integrating prior recommendations for comparing causal maps (e.g., Armstrong, et al., 2007, Chandra & Loosemore, 2010, Langfield-Smith & Wirth, 1992, Nadkarni & Narayanan, 2005, Reid, et al., 2010)), we compared maps at three levels: map, construct, and between constructs. For this purpose, we drew inferences from the qualitative interview statements (e.g., how many times interviewees referred to the linkage between project communication barriers and effective knowledge sharing) and the maps themselves (e.g., the number of constructs in the project manager’s map) to allow comparison of maps. According to the CCM literature, drawing inferences and reporting findings require artistic skills to follow causal implications reflected in the maps and information from initial and follow-up interview sessions (Tyler & Gnyawali, 2009). Therefore, besides using analytical measures (map level, construct level, between constructs level) to provide a foundation for our comparative analysis, we also remained flexible to draw upon empirical statements and underlying concepts (see Barrier Emphasis section or Table 5, for example). In summary, while the analyses involved some quantitative data, they were qualitatively driven and aimed at providing qualitative insights (e.g., Ghobadi & Ghobadi, 2013, Nelson, et al., 2000, Reid, et al., 2010). As is customary in qualitative studies, we used and report sample quotes to bring the key concerns of interviewees into life (Sarker & Sarker, 2009).

- **Map-level** analysis compared comprehensiveness and density of the maps. *Density* describes the interconnectedness of the constructs in the map; it is calculated by dividing the number of links among constructs to the number of constructs in the map (Chandra & Loosemore, 2010). High density indicates a well-understood concept by interviewees, whereas low density means a simpler and less understood concept. *Comprehensiveness* is an indicator of the depth and breadth of understanding a phenomenon; it is calculated by counting the number of constructs in a map. High comprehensiveness reflects multidimensionality of the interviewees’ viewpoint pertaining to the concept, whereas low comprehensiveness refers to limitations in perceiving the concept from different angles.

- **Construct-level** analysis compared maps in terms of centrality of constructs for each role map. *Centrality* is an indicator of how central or important a construct is to the map; it is calculated by dividing the number of direct linkages involving the construct to the total number of linkages in the map. Understanding centrality of variables across different maps is important since this measure indicates similarities and differences of the perceived importance of constructs across different
groups. We examined and compared centrality of barrier constructs in each map to understand and report how central each barrier construct was for each role.

- **Between-constructs-level** analysis examined reachability between constructs. Reachability is an indicator of the total strength of the connection between two constructs; it is calculated as the sum of the direct and indirect effects of one construct on another construct. We examined reachability between barrier constructs in each map to understand and report how each role emphasized the relationship between specified constructs (e.g., relationship between product communication barrier and effective knowledge sharing was reported to be strong in developer map). Directionality of the linkages can be shown by symbols "+" or "−", where "+" indicates two factors are positively related, whereas "−" indicates an inverse relationship.

**Conceptual Framework:** Next, to integrate the findings from the barrier items (Appendix 2) and causal maps (Figure 1-4) into more general empirical findings (Lee & Baskerville, 2003), we followed three steps. First, we carefully went through barrier constructs and their associated barrier items (Appendix 2 and Figure 1-4), removed redundant items, merged similar and relevant items, and took initial steps away from company specific jargon. For example, under barriers related to ‘team capabilities’ we merged ‘Not Enough IT Human Resources (client company)’ with ‘Lack of Experience with Software Development Companies (client representatives)’ and we renamed them as ‘Lack of IT resources and software development experience in client company’. Second, we considered core publications on knowledge sharing in the agile literature (Augustine, et al., 2005, Conboy, et al., 2010, Conboy & Morgan, 2011, McAvoy, et al., 2012, Ramesh, et al., 2010, Vidgen & Wang, 2009) to complement and further develop the conceptual framework. As an example of the impact of the agile literature, we added ‘fear of self-exposure deficiencies in development team’ based on Conboy and Morgan’s (Conboy & Morgan, 2011) reference to fears of exposing weaknesses through highly-intensive agile practices such as stand-up meetings and onsite customers. Such fears make team members feel unsafe in sharing their honest opinions and ideas. In total, six items were added. Third, to examine the application and usefulness of the framework, we presented it to software professionals at Alpha. We incrementally conducted four 1-hour evaluation sessions by incorporating the received feedback at the end of each session prior to its presentation to the next evaluators. The four sessions involved (i) one project manager and developer, (ii) one project manager, (iii) one tester and one project manager, and, (iv) one user representative, respectively. We terminated the evaluation of the framework at this point as the last two sessions only led to very minor changes and the participants found the framework comprehensive and suggested using it for managing agile projects. The evaluation sessions resulted in revised formulations of ten barriers to...
increase clarity and improve understanding. For example, ‘Neglect of non-functional requirements by development team’ was revised as ‘Neglect of non-functional requirements by stakeholders’. As a result, we came up with a list of 37 knowledge sharing barriers (Appendix 3) and a general conceptualization of knowledge sharing barriers (Figure 5).

**Results**

**Comparison of Role Maps**

To provide insights into the research question of ‘how do key stakeholders in agile software development perceive barriers to effective knowledge sharing’, we first present causal maps that resulted from our analyses of the interviews with project managers, developers, testers, and user representatives at Alpha and Beta. Figure 1, 2, 3, and 4 show the causal maps for the roles of project manager, developer, tester, and user representative. The seven constructs and their definitions are:

1. **Team diversity barriers** refer to conceptual, geographical and time differences between team members that may hinder effective knowledge sharing
2. **Team perceptions barriers** refer to attitudes and values of team members that may hinder effective knowledge sharing
3. **Team capabilities barriers** refer to knowledge and skill related issues amongst team members that may hinder effective knowledge sharing
4. **Project communication barriers** refer to communication-related issues within the project that may hinder effective knowledge sharing
5. **Project organization barriers** refer to aspects of the organization and conduct of the project that may hinder effective knowledge sharing
6. **Project setting barriers** refer to task and context related issues that may hinder effective knowledge sharing
7. **Project technology barriers** refer to technological issues that may hinder effective knowledge sharing

The transparent gray (fading) constructs and linkages exist in the overall map, but not in that particular role map. We include these fading constructs and linkages to remind readers of the overall picture that was suggested by all roles and to enable a visual comparison between single maps and the aggregate data. For example, the project manager map (Figure 1) demonstrates that project managers referred to all seven knowledge sharing barriers constructs (comprehensive map). The map’s density, which represents
interconnectedness of the constructs in the map, was 3.38 (calculated by dividing 27 (we counted the total causal links that were mentioned by project managers during interviews) by 8 (number of constructs in the map)). In terms of centrality, the most central constructs in Figure 1 were team perceptions and effective knowledge sharing constructs (0.59, 0.56), calculated by dividing the number of direct linkages involving the specified construct (as stated in the interviewees’ statements) by the total number of linkages in the map (16/27 and 15/27).

![Figure 1. Project Manager Map](image-url)
Figure 3. Tester Map
In terms of reachability, all barrier constructs in Figure 1 influenced effective knowledge sharing negatively (the bold links). Reachability, which indicates the total strength of the connection between two constructs, was highest between project setting barrier construct and effective knowledge sharing construct, suggesting that project managers put a strong emphasis on the negative impact of project setting barriers on knowledge sharing practices. Besides, some constructs influenced each other (dashed links). For example, team diversity barriers gave rise to team perception barriers (e.g., unfamiliarity of team members made them prefer work independently).

To compare maps, in terms of comprehensiveness, all maps encompass the seven barrier constructs, with the exception that the developer map does not include the team perceptions construct. The user representative map was denser (density = 4.13), compared to other maps (project manager=3.38, developer=2.57, tester=3.25). Concerning centrality measures, team perceptions and project setting were more central in the project manager map (0.59, 0.19), compared to other maps (developer=0, tester=0.38, user representative= 0.12; developers=0.06, testers=0.08, user representatives= 0.06). This is expected as project managers often are in a position to observe team perceptions and attitudes and report on them.
Team capabilities were found to be more central in the developer and user representative maps (0.28 and 0.27), compared to other maps (project managers=0.11 and testers=0.08). Developers and users are the central roles for moving development forward. Thus, they naturally place a special emphasis on this construct. We also noted that project organization was shown to be more central in the tester map (0.35) compared to other maps (project managers=0.15, developers=0.28, user representatives = 0.12). The role of testers mainly involves working with developers and sometimes project managers, thus they tend to see organizational problems within the team, while developers are busy with development and coding and project managers are focused on negotiations with client and facilitation of developers. Finally, project technology and project communication were found to be considerably more central in the user representative map (0.12, 0.36), compared to other roles. This may be because users are likely to suffer from challenging communication between the development company and themselves. The developer map was second in emphasizing project communication (0.28).

In terms of reachability measures, the strongest linkage in the project manager map was the relationship between project setting and effective knowledge sharing (0.44). This is understandable since project managers are often most involved with project setting barriers such as project budget, organizational politics, and organizational culture, which in agile development may severely impact the development team’s ability to, for example, provide demonstrations at the end of tight two-week sprints. Illustrating this linkage, one project manager stated:

“When you have a budget that is small, every single meeting you go to is costing the project. So, you have to balance how much time to spend talking to people versus how much time you spend getting things done.”

The strongest reachability measures in the developer map, tester map, and user representative map were, respectively: project communication barriers and effective knowledge sharing (0.44), project organization barriers and effective knowledge sharing (0.54), and team capabilities barriers and effective knowledge sharing (0.45). These results and reexamination of the interviewee statements suggest that close communication with end users (and not only client representatives) is important for developers to understand them better and do their jobs better. This issue may, however, be overlooked during distributed agile projects where product owners are primarily in charge of dealing with clients. As one developer stated:

“I am in charge of administrating the application after it is in production. But I haven’t actually had much chance to talk to people who are actually going to use the system on a day to day basis. Maybe that
is not the case for the [Project manager]. Because I know they did go to the [client company] a couple of
times.”

We noted that testers placed a special emphasis on organizational-related issues surrounding agile
practices such as Scrum. Specifically, the development team may attempt to adhere to the agile value of
‘individuals and interactions over processes and tools’ without following an established structure and
vision for review sessions. One tester stated:

“In the Scrums, we go around the circle and you say what you did yesterday and what you are doing
today and if you need any help. But they tend to become a place where someone will say a bit of
something and then zone out. So, Scrums more became a place to just saying things and then go back to
work.”

Finally, user representatives tended to blame the development team (i.e., project managers, developers
and testers) for lack of capability to understand users. This is problematic in agile development where
software is developed incrementally and communication problems easily may result in major reworks or
wrong estimates. As one user representative said:

“I am not quite sure how they [development company] train their people to gather knowledge and
translate it into the computer language or the system. If you ask nurses [end users], they wouldn’t give
you any straight answer ... because they haven’t any knowledge of software development and they will
probably say that is right, and afterwards when you do the testing, they’d say: oh, that is not quite what
we want.”

We summarize the centrality and reachability results in Table 3, showing which maps place the strongest
emphasis on each of the seven barrier constructs. For example, project setting is most central in the
project manager map (we noted a C for centrality in that cell). In addition, the strongest link in the project
manager map is the relationship between project setting barriers and effective knowledge sharing (we
noted an R for reachability in that cell).

<table>
<thead>
<tr>
<th>Table 3. Centrality and Reachability Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructs</td>
</tr>
<tr>
<td>1. Team Diversity Barriers</td>
</tr>
<tr>
<td>2. Team Perceptions Barriers</td>
</tr>
<tr>
<td>3. Team Capabilities Barriers</td>
</tr>
<tr>
<td>4. Project Communication Barriers</td>
</tr>
<tr>
<td>5. Project Organization Barriers</td>
</tr>
</tbody>
</table>

Overlaying the centrality and reachability results, Table 3 suggests that project setting, project communication, project organization, and team capabilities barriers are the key foci of the project manager, developer, tester, and user representative map, respectively.

**Barrier Emphasis across Roles**

To add further insights on comparative analysis, Table 4 demonstrates how each role emphasized the seven barrier constructs. The Table shows how the underlying concepts in each construct (see Appendix 2 for constructs and their underlying concepts) are highlighted by each role. For example, the first row of Table 4 shows that out of 6 concepts within team diversity barriers, project managers, developers, testers, and user representatives raise 5, 3, 2, and 3 concepts, respectively. The information in Table 4 concurs with the findings in Table 3; the most emphasized barrier construct by each role is the same as the bold items in Table 3. Below, we provide a detailed explanation of how each role instantiated each construct.

**Table 4. Emphasis on the Underlying Concepts of Barriers Constructs**

<table>
<thead>
<tr>
<th>Constructs</th>
<th>Project Managers</th>
<th>Developers</th>
<th>Testers</th>
<th>User Representatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Team Diversity Barriers /6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2. Team Perceptions Barriers /8</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3. Team Capabilities Barriers /10</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>4. Project Communication Barriers /11</td>
<td>1</td>
<td><strong>6</strong></td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5. Project Organization Barriers /12</td>
<td>5</td>
<td>3</td>
<td><strong>8</strong></td>
<td>3</td>
</tr>
<tr>
<td>6. Project Technology Barriers /4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>7. Project Setting Barriers /9</td>
<td><strong>8</strong></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**1. Team Diversity Barriers:** Project managers refer to 5 out of 6 empirically observed concepts within the team diversity construct whereas other roles refer to fewer concepts (3, 2, and 3). This is understandable since project managers are in a position to observe team diversity across different groups and its consequences on team behaviors. Illustrating this, one project manager stated:

“We had a huge amount of discrepancy in terms of people’s knowledge. We had someone who knows the [domain knowledge] over five years. We had people who had never worked on [this area].”

**2. Team Perceptions Barriers:** Project managers and testers have very different perceptions compared to user representatives. They raise team-related challenges (e.g., low levels of focus on project), and in one

case they also blame client attitudes (e.g., inappropriate assumptions made by the client). As one project manager stated:

“They had certain expectations about what they could get with the very small budget they had ... It turned out they [client representatives] were not telling us these ... Because we are agile and flexible they assumed we are just going to do that”.

However, user representatives point to the relationship between project technology and team perceptions to explain how inappropriate methodology (agile methodology) makes the development team inflexible to innovation and change. In contrast, the development team tends to avoid raising challenges related to agile methodology. One user representative said:

“I found that the format of agile restricts innovations and restricts knowledge sharing. If they [development team] don’t have a story written it seems to inhibit their initiative to actually be proactive and do something about it. And I’ve been in meetings where something glaring has come up, and I’d say why this wasn’t done, why wasn’t this raised, and someone goes: we didn’t have a story.”

3. Team Capabilities Barriers: This construct includes both technical and social team capabilities. A close look at the concerns of the different roles shows that the development team focused on technical barriers (e.g., unfamiliarity with the coding language). As one developer stated:

“It was Ruby and Rails application. I hadn’t used it before. Well, I would probably be the main barrier because I didn’t know the language.”

However, user representatives highlighted the importance of social capabilities for achieving effective knowledge sharing.

“The [development company] needs to understand what [our business] is about, and what people doing [this business] look like. Then they will be able to build more effective communication with us.”

In addition, there was not much consistency in how project managers, developers, and testers instantiated barriers associated with team capabilities. Testers were influenced by the stories that are often provided by project managers and they work with developers, so they raised issues about developer and project manager capabilities. In contrast, project managers were critical towards the capabilities of user representatives (e.g., lack of experience with software development companies).

4. Project Communication Barriers: Out of the 11 concepts within the project communication barriers’ construct, project managers, developers, testers, and user representative raised 1, 6, 4, and 6 concepts, with different roles referring to different concepts. Developers blamed user representatives (e.g., improper
check of functionalities with end users), while user representatives criticized the development team for their communication-related limitations (e.g., passive engagement with the client). Both developers and testers seemed willing to have more face to face communication with the user representatives, while project managers believed it was better if they were in charge of communication with their client. One project manager stated:

“*We have a very small set of people who understand what the clients want and can create a common vision rather than everyone else having their own opinion about how that works.*”

Not surprisingly, project managers did not raise many concerns (1 concept) regarding project communication barriers.

5. **Project Organization Barriers:** Testers refer to 8 out of 12 empirically observed concepts within this construct, which is more compared to other roles (5, 3, and 3). It is understandable that user representatives do not emphasize this category, since the development team is mainly involved in the organization of project activities. Considering the development team, developers, testers and project managers do not refer to the same concerns. Testers mainly refer to project organization barriers, likely because they are highly influenced by developers (test of stories) and project managers (assigning insufficient resources) and therefore tend to emphasize organizational challenges within the team. However, developers are often busy with coding and project managers are mainly focused on negotiations with user representatives and facilitating developers. One tester said:

“*We were at a peak, five developers to one tester. So you’d have a large amount of work coming through at any point of time. So you can have a basic understanding, but not a deep understanding.*”

6. **Project Technology Barriers:** User representatives placed great emphasis on all the concepts within this construct (4 out of 4), whereas other roles were concerned about one concept. User representatives questioned the employed technology broadly, whereas the development team did not. The development team raised concerns about collaboration technologies. As one project manager stated:

“*We did not have clear defined templates or ways to use Confluence. It doesn’t have a great set of collaborative tools, at least the version that we use.*”

Only one developer raised challenges related to the agile methodology, and this developer had only been working on agile projects for seven months.

7. **Project Setting Barriers:** This barrier included 9 concepts, and project managers pointed to 8 of them, whereas other roles referred to only 2 concepts out of 9. Hence, project managers were very concerned
about challenges that project setting can bring to agile development. This is understandable because their role requires them to deal with project setting issues. For example, compared to other roles project managers are in a very good position to see how legacy systems, cultural issues, organizational politics, and budget challenges impact development practices. As one project manager stated:

“They [client company] had a small database that was developed by someone who was not part of the team. So, their understanding of certain details and requirements of the project was unclear and even misleading.”

**Conceptual Framework of Barriers**

Our final framework consists of a list of 37 knowledge sharing barriers (Appendix 3) and a general conceptualization of knowledge sharing barriers (Figure 5).

---

**Figure 5. Conceptualization of Barriers to Effective Knowledge Sharing in Agile Software Teams**
Focusing on providing a conceptualization for knowledge sharing barriers, Figure 5 reveals three types of barriers (team factors, process factors and contextual factors) and seven specific barriers (team diversity, team perceptions, team capabilities, project communication, project organization, project setting, project technology) based on the 37 barrier items in Appendix 3 (as detailed in the Data Analysis section).

By providing a general conceptualization of knowledge sharing barriers in agile teams, Figure 5 provides a fertile ground for future research that specifically focuses on establishing theoretical and empirical models of the interrelations between barrier constructs. Similar to how Leavitt’s model depicts all possible interactions between the four components of people, structure, task, and technology (Lyytinen & Newman, 2008), we show linkages between all barrier constructs as thin lines to highlight possible relationships that should be examined in further investigation and model building.

Discussion

Contributions to Theory

The objective of this study was to explore the similarities and differences in how key actors perceive barriers to effective knowledge sharing in agile development. Framing our research question as ‘how key stakeholders in agile software development perceive barriers to effective knowledge sharing?’ we first built on organizational role theory (Katz & Kahn, 1978) and previous systems development literature (Robey, et al., 2001) to focus on the recurrent actions performed by the four key roles of ‘project manager’, ‘developer’, ‘tester’, and ‘user representative’. Informed by the literature on knowledge sharing within software teams in general (Ghobadi & D'Ambra, 2013, Joshi, et al., 2007, Oshri, et al., 2008) and agile teams in particular (Chan & Thong, 2009), we employed CCM to provide qualitative insights into how the four roles perceive barriers to effective knowledge sharing. CCM enabled us to (i) evoke empirical and generalized constructs of knowledge sharing barriers, (ii) identify the complex chains of arguments that emerged from the interview statements as cognitive maps (Figure 1-4), and (iii) reveal cognitive similarities and differences in how the observed team roles perceive knowledge sharing barriers. The application of CCM added richness and structure to the findings and allowed us to offer three theoretical contributions to the literature.

First, we have presented empirically grounded models of the diverse perspectives on knowledge sharing barriers across four key roles in agile teams. These findings represent generalizations from theory (organizational role theory and models of knowledge sharing) and description (interviews statements) to description (Lee and Baskerville, 2003). As summarized in Table 5, barriers were, despite certain
similarities, perceived differently across the roles. Specifically, as discussed in detail in the Result section we found notable variation in how roles emphasized the seven barriers (3rd point in Table 5). In addition, different roles referred to different instances of barriers constructs (4th point in Table 5). The causal models (Figures 1-4) and the summary in Table 5 support prior studies that sees roles as patterns of actions and interactions that influence the course of events (Galletta & Heckman Jr, 1990, Zigurs & Kozar, 1994), and, as such they may support continued efforts to explicate and promote roles in software development contexts (Mähring, et al., 2008). Some of these results concur with the previous studies that link different roles to tasks performed (Huisman & Iivari, 2006). For example, it is understandable that user representatives consider developers’ lack of knowledge about their field a barrier to knowledge sharing, while developers, in turn, may perceive political challenges within the client company, reflected in frequent change of IT representatives, as major barriers. In addition, our results shed light on important perceptual differences that are unique to agile contexts, as agile values and principles (e.g., welcoming change and measuring progress based on the high-quality software developed) may reinforce communication-related difficulties (Persson, et al., 2012) and inconsistencies (Boehm, 2002, Pikkarainen, et al., 2008, Turk, et al., 2005). For example, user representatives interpreted developers’ inflexibility in welcoming change as a barrier to innovation and effective knowledge sharing, while developers considered users to have inappropriate expectations to how flexible developers can and should be, thus inhibiting effective discussion up front. In another example, although the development team tried to adhere to the agile principle of ‘measure progress based on the high-quality software developed’, they tended to overlook the importance of coaching, mentoring and voluntary contributions in performance evaluations (Conboy, et al., 2010). While such perceptual differences are inevitable, awareness of how they manifest across key roles in agile teams reveal new insights into the limited IS research that focuses on organizational role theory and interaction between roles (Barki & Hartwick, 2001, Mähring, et al., 2008, Zigurs & Kozar, 1994).

Table 5. Empirical Findings on Barriers to Effective Knowledge Sharing in Agile Software Teams

<table>
<thead>
<tr>
<th>Summary of Empirical Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. All roles highlighted the seven barrier constructs, with the exception that developers did not refer to barriers associated with team perceptions.</td>
</tr>
<tr>
<td>2. User representatives revealed more cognitive linkages between constructs (density of 4.13), compared to other roles (project manager (3.38), developer (2.57), and tester (3.25)).</td>
</tr>
<tr>
<td>3. There was variation in how roles emphasized the seven barriers, for example:</td>
</tr>
<tr>
<td>o Project managers put primary emphasis on ‘project setting’ barriers</td>
</tr>
<tr>
<td>o Developers put primary emphasis on ‘project communication’ barriers</td>
</tr>
<tr>
<td>o Testers put primary emphasis on ‘project organization’ barriers</td>
</tr>
<tr>
<td>o User representatives put primary emphasis on ‘team capabilities’ barriers</td>
</tr>
<tr>
<td>4. There was variation in how roles instantiated each barrier, for example:</td>
</tr>
</tbody>
</table>

For **team perceptions barriers**, project managers, developers and testers raised concerns about their own limited focus on the projects and inappropriate assumptions made by the client, whereas user representatives were mainly concerned with the inflexibility of the development team related to innovation and change.

- For **team capabilities barriers**, developers focused on their own limited technical capabilities, whereas user representatives questioned social skills of development team. In addition, there was not much consistency in how developers, testers, and project managers instantiated barriers associated with team capabilities.

- For **project communication barriers**, developers blamed user representatives for improper check of functionalities with end users, whereas user representatives criticized the development team for passive engagement.

- For **project technology barriers**, user representatives questioned the employed technology broadly (in particular appropriateness of the agile methodology), whereas the development team did not raise challenges associated with the agile methodology.

Second, we drew upon existing models of knowledge sharing and interview statements to develop a conceptual framework of knowledge sharing barriers in agile teams (Figure 5 and Appendix 3). Following Lee and Baskerville (2003), this framework represents generalization from theory and description to theory, hence contributing to moving agile software development forward as a scientific discipline (Dingsøyr, et al., 2012, Dybå & Dingsøyr, 2008). This is arguably a more comprehensive and systematic conceptualization of knowledge sharing barriers compared to what has previously been published in both the agile literature (Dingsøyr & Hanssen, 2003, Korkala & Abrahamsson, 2007) and the general software literature (Joshi, et al., 2007, Pee, et al., 2010). By presenting a comprehensive picture of knowledge sharing barriers (Figure 5, Appendix 3), our study covers some general software risk items (Boehm, 1991) and classic mistakes in agile practices (McConnell, 2010). For example, barriers such as lack of motivations, unrealistic and estimation and expectations, and insufficient planning, are also mentioned as classic mistakes in rapid development (McConnell, 2010). However, our framework is different in a number ways. First, it is distinctly focused on knowledge sharing barriers, rather than on general performance outcomes. Second, it delves into some unique challenges of agile development that are not mentioned in similar frameworks (McConnell, 2010), e.g., by specifying barriers such as ‘lack of purpose and proper organization in Scrum review meetings’, ‘making decision in development without consulting client (due to tight sprint schedules)’, and ‘lack of communication of agile time requirements with client up front’. Moreover, the framework merges similar items, removes redundant items, and has taken steps away from company-specific language and jargons (e.g., merging ‘omitting necessary tasks from estimates’, and, ‘overly optimistic schedules’). Accordingly, the framework in Figure 5 and Appendix 3 opens for qualitative and quantitative studies of the dynamic interplay between knowledge sharing barrier constructs, a line of investigation that has not been sufficiently addressed in previous research (e.g. (Ghobadi & D’Ambra, 2013, Joshi, et al., 2007, Pee, et al., 2010)).

Third, we add to the promising literature on CCM (Laukkanen, 1994, Laukkanen & Päivi, 2013). According to Laukkanen and Päivi, methodological discussions of CCM are relatively rare. For example,
there are no firm and standard guidelines on measurement indicators in CCM. Accordingly, we adapted existing guidelines and examined maps at three levels of analysis—map, construct, and between constructs. We also went beyond existing guidelines by drawing upon empirical statements to discuss how roles instantiated barriers and how they emphasized each barrier construct. Comparing this to prior studies (Chandra & Loosemore, 2010, Nadkarni & Narayanan, 2005), we suggest our methodological approach as a systematic and innovative approach to CCM that can be used by IS as well as non-IS researchers.

Finally, we employed polar sampling (Eisenhardt & Graebner, 2007) of high-performing and low performing projects to secure variance across selected projects and add insights related to project outcomes. Comparing barriers between the two high performing projects (HA, HB) and the two low performing projects (LA, LB), we observed a total of 8, 15, 30, and 20 knowledge sharing barriers items for projects HA, HB, LA, and LB. Hence, the high performance projects experienced fewer knowledge sharing barriers than the low performing. Recognizing the relationship between knowledge sharing barriers and performance outcomes (high performance projects experienced fewer knowledge sharing barriers than the low performing), future research may focus on how high performing and low performing projects differ in the way they address knowledge sharing barriers and if there is any relationship between patterns of employed resolution techniques and performance outcomes.

**Managerial Implications**

Personal and environmental demands impose different role perceptions and behaviors (Barki & Hartwick, 2001, Newman & Robey, 1992, Robey, et al., 2001). While these differences are natural, being aware of such variations is critical to the success of software projects. Software organizations should consider educating their development teams to have a better understanding of knowledge sharing barriers and differences in perceptions across key stakeholders (Figures 1-5, Appendix 3). To support collective mindfulness (McAvoy, et al., 2012), project managers are encouraged to involve team members in the analysis of barriers against the model provided in Figure 5 and the 37 identified barriers. Such collective efforts may assist team members in sharing mental models to proactively address conflicts and cultivate team member collaboration. In addition, managers should understand the different ways barriers may become manifest (Barrier Emphasis across Roles section). For example, supporting the previous literature on software team characteristics (Siau, et al., 2007), we found inherent differences in the manifestation of team capabilities barriers across roles. Developers tended to focus on technical aspects of team capabilities, whereas user representatives highlighted the importance of social capabilities. Taking into
account such factors, managers may revisit recruiting, training, and human resource management for agile teams.

To address knowledge sharing barriers (Appendix 3), approaches need to consider context and they change within the same company over time, due to factors such as current characteristics of the development team, client characteristics, and project complexity. The agile literature points to a number of approaches for overcoming knowledge sharing barriers (Conboy, et al., 2010, Ramesh, et al., 2010, Vidgen & Wang, 2009), e.g., Vidgen and Wang (2009), recognizing practices such as retrospectives and pair programming as well as the enabling influence of explorations to learn about new areas and share them with the team (Conboy, et al., 2010). The knowledge management literature refers to several approaches such as “community forums” and “use of space” to facilitate knowledge sharing (Earl, 2001). Moreover, the software development literature offers suggestions for improving knowledge sharing and overcoming barriers (Bjørnson & Dingsøyr, 2008, Mathiassen & Vogelsang, 2005), e.g., Mathiassen and Vogelsang (2005) discuss network and networking approaches to knowledge sharing—network approaches emphasize the use of technology for sharing knowledge, while networking approaches focus on creating trust and collaboration among those involved in software development. Before any approaches are enacted, managers should be aware of possible inhibiting impacts. Conboy and Morgan (2011) showed, for example, that daily stand-ups can reduce the amount of team members’ time to share and discuss ideas, and, a case involving a Jamaican-Indian software team revealed how boundary objects involving definitional control and the subsequent redistribution of power led to emerging cross-cultural differences and in turn inhibited knowledge sharing (Barrett & Oborn, 2010). With respect to the dynamic views of knowledge (Choi & Lee, 2002, Hansen, et al., 1999), managers of knowledge sharing practices should consider the inherent characteristics of knowledge, e.g., whether knowledge is mainly tacit or explicit (Bohn, 1994, Choi & Lee, 2002). The knowledge management literature also distinguishes between system-oriented strategies focused on creation, storage, sharing, and use of knowledge via information technology (mainly involving explicit knowledge), and human-oriented strategies emphasizing dialogue through social and interpersonal networks (involving tacit knowledge).

**Limitations and Future Directions**

We employed a qualitative methodology, CCM, in a little understood domain: perceptual differences to knowledge sharing behaviors in agile teams. Although CCM enabled us to develop cognitive maps that can serve as guidance for further qualitative or quantitative inquiry, the approach was based on a limited sample size. Moreover, our study was conducted in two medium size organizations in Australia. We
recommend that large-scale empirical studies be undertaken to validate, modify, or extend the presented causal maps. In doing so and in particular for larger samples we encourage using available software for comparative causal mapping such as CMAP3 or Cognizer (Clarkson & Hodgkinson, 2005, Laukkanan & Päivi, 2013). Researchers may investigate approaches and interventions that allow for more fruitful conversations across different roles and possibly map the identified barriers to process and product-related uncertainties respectively (Mathiassen & Pedersen, 2008).

Acknowledgment

We would like to thank the editors and reviewers for their constructive feedback and comments.

References


Dingsøyr, T. & Hanssen, G. (Year) Extending agile methods: postmortem reviews as extended feedback, Fourth International Workshop on Learning Software Organizations, Chicago, IL, USA.


Holsti, O.R. (1969) Content analysis for the social sciences and humanities, Addison-Wesley, Don Mills, ON.


Melnik, G.& Maurer, F. (Year) Direct verbal communication as a catalyst of agile knowledge sharing. Agile Development Conference, Salt Lake City, UT, United States.


Paetsch, F., Eberlein, A.& Maurer, F. (Year) Requirements engineering and agile software development, IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises Linz, Austria.


Appendix 1. Interview Guide

- In this project, how important was “effective knowledge sharing” among team members?
- In what ways was knowledge sharing practiced?
- During the project, how satisfied you were by the knowledge sharing practices among team members?
- In what ways did knowledge sharing help the project achieve its goals?
- Which problems did you notice in achieving effective knowledge sharing among team members?
- What were the key enablers of effective knowledge sharing?
- What were the key barriers to effective knowledge sharing?
- Now, I have noted a number of barriers to effective knowledge sharing in this project. Can you please have a look and sort them for me based on their level of importance.

(This question was asked as recommended by cognitive mapping literature to allow us find a better sense of the cognitive importance of concepts and constructs (Tyler & Gnyawali, 2009))

- Is there anything else that you would like to mention that we did not cover?

### Appendix 2: Barrier Constructs and Items

<table>
<thead>
<tr>
<th>1. Team Diversity (6 concepts)</th>
<th>2. Team Perceptions (8 concepts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different Background (e.g., education background, experience, balance of technical knowledge power)</td>
<td>Lack of Motivation (development team)</td>
</tr>
<tr>
<td>Unfamiliarity of Team Members (not having prior work experience together) (development team)</td>
<td>Not Being Flexible to Innovation (development team)</td>
</tr>
<tr>
<td>Physical Distance between Team Members</td>
<td>Cultural Fears regarding Estimation (development team)</td>
</tr>
<tr>
<td>Different Languages (development team)</td>
<td>Tendency to Work Independently (development team)</td>
</tr>
<tr>
<td>Different Languages (development and client team)</td>
<td>Low-Levels of Trust between Team Members</td>
</tr>
<tr>
<td>Time Difference (between development and client company)</td>
<td>Low Levels of Focus on Project (development team)</td>
</tr>
<tr>
<td></td>
<td>Perceiving the Organizational Importance of the Project to be Low (development team)</td>
</tr>
<tr>
<td></td>
<td>Inappropriate Assumptions (made by client representatives)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Team Capabilities (10 concepts)</th>
<th>4. Project Communication (11 concepts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unfamiliarity with the Coding Technology (development team)</td>
<td>Lack of Face-to-Face Communication with Client (development team)</td>
</tr>
<tr>
<td>Unfamiliarity with the Business Domain Knowledge (development team)</td>
<td>Client Unavailability for Communication</td>
</tr>
<tr>
<td>Unfamiliarity with the Employed Collaborative Technologies (development team)</td>
<td>Slow Turn Around (client representative)</td>
</tr>
<tr>
<td>Writing Unclear Stories (development team)</td>
<td>Passive Engagement with the Client (development team)</td>
</tr>
<tr>
<td>Lacking IT Knowledge (client representative)</td>
<td>Not Putting the Client in the Loop of Related Communications (development team)</td>
</tr>
<tr>
<td>Not Having Sufficient Understanding about the Technical Requirements (development team)</td>
<td>Not Communicating Agile Time Requirements to the Client (development team)</td>
</tr>
<tr>
<td>Low-Levels of Social Skills (development team)</td>
<td>Not Sharing Information about Client Feedback with the Development Team (Project Management)</td>
</tr>
<tr>
<td>Not Understanding Client Business (development team)</td>
<td>Miscommunication Within the Client Company (IT department)</td>
</tr>
<tr>
<td>Lack of Experience with Software Development Companies (client representatives)</td>
<td>Miscommunication about Project Structure and Responsibilities (between development and client company)</td>
</tr>
<tr>
<td>Not Enough IT Human Resources (client company)</td>
<td>Improper Check of Functionalities with End Users (client representative)</td>
</tr>
<tr>
<td></td>
<td>Sharing Complex Technical Information with the Client Representative (development team)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. Project Organization (12 concepts)</th>
<th>6. Project Technology (4 concepts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Much Planning beforehand (development team)</td>
<td>Lack of High-Quality Collaborative Technologies</td>
</tr>
<tr>
<td>Assigning Few Testers Compared to the Number of Developers</td>
<td>Lack of Defined Templates for Collaboration Tools</td>
</tr>
<tr>
<td>Multi-Tasking (development team)</td>
<td>Lack of a Good Prototype for Communicating Requirements</td>
</tr>
<tr>
<td>Lack of Continuity (development team)</td>
<td>Inappropriate Methodology (agile)</td>
</tr>
<tr>
<td>Several IT Representative Roles (client company)</td>
<td></td>
</tr>
<tr>
<td>Unrealistic Estimations by development team (development team)</td>
<td></td>
</tr>
<tr>
<td>Making Decision without Consulting the Client (development team)</td>
<td></td>
</tr>
<tr>
<td>Tight Schedule</td>
<td></td>
</tr>
<tr>
<td>Long Split Sprints</td>
<td></td>
</tr>
<tr>
<td>Providing Dummy Data for Test (client company)</td>
<td></td>
</tr>
<tr>
<td>Scrum Review Structures (less objective while sharing updates)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Project Setting (9 concepts)</td>
<td></td>
</tr>
<tr>
<td>Domain-Specific Project</td>
<td></td>
</tr>
<tr>
<td>Multi-Dimensional Project (software and infrastructure sides)</td>
<td></td>
</tr>
<tr>
<td>Complex Business Rules</td>
<td></td>
</tr>
<tr>
<td>Small Budget Project</td>
<td></td>
</tr>
<tr>
<td>Existence of Legacy Systems</td>
<td></td>
</tr>
<tr>
<td>Not being able to Choose Team Members for the Project (nature of the project)</td>
<td></td>
</tr>
<tr>
<td>Commercial Culture (development company)</td>
<td></td>
</tr>
<tr>
<td>Agile Versus Non-Agile Cultures (client and development company)</td>
<td></td>
</tr>
<tr>
<td>Bureaucratic Organization (client company)</td>
<td></td>
</tr>
</tbody>
</table>
## Appendix 3: Finalized Barrier Constructs and Items

<table>
<thead>
<tr>
<th>Team Diversity refers to conceptual, geographical and time differences between team members that may hinder effective knowledge sharing.</th>
<th>Team Perceptions refers to attitudes and values of team members that may hinder effective knowledge sharing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Different speaking languages among members</td>
<td>1. Lack of motivation, focus and adaptability in development team</td>
</tr>
<tr>
<td>2. Different working and disciple-related backgrounds among members</td>
<td>2. Inappropriate assumptions about project scope made by client (due to the development team’s flexible agile-related approach)</td>
</tr>
<tr>
<td>3. Different time zones and physical distance between members</td>
<td>3. Fear of self-exposure to technical and agile skills deficiencies in development team</td>
</tr>
<tr>
<td>4. Lack of prior joint working experience in development team</td>
<td>4. Performance evaluation based on technical achievements</td>
</tr>
<tr>
<td>5. Stakeholder neglect of nonfunctional requirement</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team Capabilities refers to knowledge and skill related issues amongst team members that may hinder effective knowledge sharing.</th>
<th>Project Communication refers to communication-related issues within the project that may hinder effective knowledge sharing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Insufficient understanding of business domain and context</td>
<td>1. Inadequate client availability and participation</td>
</tr>
<tr>
<td>2. Unfamiliarity with development and collaboration technologies</td>
<td>2. Lack of communication of agile time requirements with client up front</td>
</tr>
<tr>
<td>3. Insufficient and ambiguous requirements</td>
<td>3. Lack of concurrence within client team</td>
</tr>
<tr>
<td>4. Inadequate social skills</td>
<td>4. Product owner lack of sharing client feedback with development team</td>
</tr>
<tr>
<td>5. Lack of familiarity with agile values and principles</td>
<td></td>
</tr>
<tr>
<td>6. Lack of IT resources and working experience with software companies in client company</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Organization refers to aspects of the organization and conduct of the project that may hinder effective knowledge sharing.</th>
<th>Project Technology refers to technological issues that may hinder effective knowledge sharing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Tight sprints schedule with little time for interaction</td>
<td>1. Lack of using high quality collaboration technologies and processes in development team</td>
</tr>
<tr>
<td>2. Inadequate planning and organization in agile practices</td>
<td>2. Lack of a good prototype to communicate requirements between stakeholders</td>
</tr>
<tr>
<td>3. Multitasking and lack of continuity in development team</td>
<td>3. Employing agile methodology without planning up front</td>
</tr>
<tr>
<td>4. Inadequate planning and insufficient documentation</td>
<td>4. Prioritization of requirements based on one-dimensional thinking</td>
</tr>
<tr>
<td>5. Making decisions in development without consulting client (due to tight sprints schedules)</td>
<td></td>
</tr>
<tr>
<td>6. Frequent change of IT representatives in client company</td>
<td></td>
</tr>
<tr>
<td>7. Centralized decision making</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Setting refers to task and context related issues that may hinder effective knowledge sharing.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Complex and domain specific project</td>
<td></td>
</tr>
<tr>
<td>2. Small budget agile project with limited room for interaction</td>
<td></td>
</tr>
<tr>
<td>3. Dependence on existing or legacy technology</td>
<td></td>
</tr>
<tr>
<td>4. Inability to choose development team members</td>
<td></td>
</tr>
<tr>
<td>5. Different approaches to agility between development and client company</td>
<td></td>
</tr>
<tr>
<td>6. Profit focused culture in development company</td>
<td></td>
</tr>
<tr>
<td>7. Bureaucratic and centralized organizations</td>
<td></td>
</tr>
</tbody>
</table>

**Bold** items are specifically agile-related.

*Italicized* items were added after incorporating material from the agile development literature.