在敏捷软件团队中有效知识共享的障碍

(Perceived barriers to effective knowledge sharing in agile software teams)

作者: 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, GA303302, USA, email: lmathiassen@ceprin.org

摘要: 虽然文献提供了几个框架，解释在软件开发团队中知识共享的障碍，但鲜为人知的是团队成员如何看待这些障碍的差异。基于四个软件项目进行多案例研究，我们研究项目经理，开发人员，测试人员和用户代表如何看待敏捷开发中的障碍和有效的知识共享。构建因果图，我们构建了因果图的四个角色并确定他们在图构建和因果联系的重叠和分歧。结果表明，尽管四种角色有一定的相似性，但如何看待和强调知识共享的障碍并不同。项目经理把主要重点放在项目设置障碍上，而开发人员，测试人员和用户代表主要关注的分别是项目沟通，项目组织和团队能力的障碍。整合四个因果图和敏捷的参考文献，我们提出了一个用七种知识共享的障碍和37个具体的障碍的概念性框架。我们认为，需要在软件团队架设沟通的桥梁并创造共同的理解，我们将不同的角色关注点纳入考量的关键。最后，我们讨论了在敏捷团队和软件团队中更普遍的知识共享。

（While the literature offers several frameworks that explain barriers to

knowledge sharing within software development teams, little is known about differ-

ences in how team members perceive these barriers. Based on an in-depth multicase 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 mapping, 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 knowledgesharing 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.）

引言：软件开发是一个合作的过程，成功取决于有效的知识共享，这是最真实的敏捷开发。将重点放在沟通，涉及到的不同利益相关者通过普遍的实践，如联合应用设计会议和客户焦点小组，从而支持敏捷实践的软件开发。敏捷宣言列出了十二个原则，如“专注于客户”，“定期合作”，“面对面的沟通团队”和“有定期团队反思”，本质上都需要团队成员之间采取有效的知识共享的做法。

然而，敏捷团队中的知识共享是具有挑战性的。例如，增加用户的动机，它可以是具有挑战性的，以分享特定的业务与开发人员的知识，管理跨功能和各种社会身份参与软件开发，和鼓励和促进团队成员之间的隐性知识共享。此外，在动态的商业环境中已经看到许多软件组织采用的方法，融合了敏捷原则与分布式开发的概念，兼顾两者的好处，但是却增加了知识共享的敏捷团队面对的挑战。

虽然在研究软件团队中的知识共享的障碍已经进行了努力，但在一般的敏捷团队，这些障碍可能无法解决，且没有考虑到所涉及的利益相关者的关注点。事实上，由于不同的角色期望和沟通的差距，软件团队成员可能有感性的差异，如项目的成功，项目失败，风险因素和适当的发展方法的问题。同样，不同的角色，如管理人员，开发人员，测试人员和用户的代表可能有感性的差异，无法有效的知识共享。例如，用户可能会认为，开发人员缺乏他们的领域知识是知识共享的一个障碍。反过来开发商可能有政治问题和不一致的用户描述的功能要求是主要的障碍。因此，开发人员可以推迟与用户的密切互动，直到他们谈判达成一项协议。关于整体项目的目标和主要要求，用户可能会反过来链接上开发人员繁忙的时间表，保持与他们互动，除非开发商采取主动。感觉和期望可能存在差异，特别是存在于团队中，坚持极端的奉献精神或解释敏捷的价值观和原则加强与沟通有关的困难并不一致。例如，用户可能希望开发商代表要不断开放新系统的要求，从而在欢迎变化这里存在一个屏障，有效的知识共享实践解读开发商“刚性”。然而，开发商可能不同意；认为除了在项目控制和问责制一定水平的重要性，欢迎变化要求中后期发展可能迫使他们加班，矛盾的敏捷原则，个人的价值超过过程。因此，开发人员可能会考虑用户代表没有注意到指定的要求，尽早有效的知识共享的一个关键障碍。

我们认为，这些感性的差异，必须考虑到创建团队成员之间的共享理解为一个关键，在组织中的性能。不同的看法可能会导致不同的行动，以克服障碍和不满意的结果，最终可能会引起利益相关者之间的冲突。显然，这样的行为建立非生产性的沟通做法和不健康的利益相关者群体之间的关系，他们可能会导致项目的延迟和小于最佳的软件解决方案。因此，利益相关者必须探索在感知上的差异，并达成协议的障碍，有效的知识共享的做法。事实上，过程中尝试和欣赏这种差异是软件团队减少共享他们所面临的障碍，知识的必要的第一步。

在一个开阔的观点对知识共享的软件团队一般的敏捷团队在特殊障碍称尽管不同角色的知觉差异仍然相对未开发的。事实上，一个系统的调查的软件团队成员关于知识共享的障碍的感性差异，还没有得到显着的关注。在这种背景下，我们采用定性的方法，比较因果图（CCM）对数据进行分析，从四的敏捷团队在两软件公司。区分“项目经理”、“开发人员”、“测试者”和“用户代表”的角色，我们构建了敏捷团队中的知识共享障碍的角色具体的因果图。结果表明，尽管有一定的相似性，跨映射，四个角色不同，他们如何看待和强调的障碍。整合的因果地图与敏捷的文献，我们也提出了一个概念框架的障碍，有效的知识共享敏捷团队，这些见解提供了宝贵的贡献，目前的理论知识共享敏捷团队和软件团队更普遍。

（Software development is a collaborative process where success depends on effective knowl-edge sharing where great emphasis is placed on communication involving diverse stakeholders through practices such as joint-application design sessions and customer focus groups. Not surprisingly, to support agile practices in software development, the Agile Manifesto lists twelve principles, 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. For example, it can be challenging to increase users’ motivation to share specific business knowledge with developers, to manage cross-functionality and the variety of social identities involved in software development and to encourage and facilitate sharing tacit knowledge among team members. 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.This has, however, increased the knowledge-sharing challenges agile teams face.While efforts have been undertaken to study barriers to knowledge sharing in software teams in general and agile teams in particular, these barriers may not be addressed without taking into account

the concerns of the involved stakeholders. In fact, because of 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. 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 that developers’ lack of knowledge about their field is 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 and inconsistencies. For example, user representatives may expect developers to be constantly open to new system requirements, 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. 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 that these perceptual differences must be taken into account to create shared understanding among team members as a key to performance in organizations. 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 behaviours 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 ofexploring 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 generaland agile teams in particular, 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) 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.

理论背景：现存的文献表明，一些试图了解知识共享的障碍或驱动程序内的软件团队。针对一般的软件团队，joshiet铝。用联结主义认识论的角度揭示了关键作用源的可信度和传播范围在塑造内部知识转移戈巴第团队。在另一项研究中，描述和安布拉的动态结果，意味着与边界的相互依存关系驱动同时合作和竞争行为，反过来影响优质知识在跨职能团队的共享软件。

转向的敏捷团队，现存的文献表明几个过程和语境相关因素推动知识共享诸如轻质验风格和充分的文献，敏捷的文献也承认动机相关的因素，合作的因素和能力相关的因素的驱动影响。

虽然一般的软件开发文学为我们提供了更坚实的理论基础，在这方面，敏捷的文献仍然是零散的有限分析的基础上，并有助于严谨的理论观点。在敏捷开发中的知识共享研究的空白是令人惊讶的，给予高度重视沟通和合作。此外，以前的研究很少关注特定的团队方面，如感知差异，知识共享在敏捷团队中的障碍，从而阻碍了对敏捷团队特征的严谨理解的发展。有了这个背景，我们设计了我们的研究，根据现有的解释，在软件团队中的知识共享的一般和敏捷团队，特别是探索知觉差异知识共享的障碍，在敏捷团队的关键角色。我们借鉴了组织的作用理论，它的根源在社会学，并提供了解个人在工作中的行为和态度的见解。该理论认为，在一个特定的工作角色的行为，指的是一组不同的活动，是组织需求的结果，社会需求和个人需求的作用。因此，一个组织的角色，由职位名称和指定的工作描述，包括预期的信念和依附于它的行为。这一理论的重点是连接的信念和行为的作用，因此，与我们的兴趣在理解在敏捷开发中的知识共享障碍的角色之间的差异。我们专注于软件开发项目中的四个关键角色，了解各种的观点和行为。具体地说，在以前的关键作用和它们之间的相互作用研究的基础上，我们针对四个角色的项目经理，开发人员，'tester '和'用户代表，涵盖软件开发中进行反复的动作。

总结，通知组织的作用理论和文献知识共享软件团队在一般和敏捷团队，特别是我们的研究目标是以下：（I）探索的相似性和差异如何四个关键角色-项目经理，“开发人员”，“测试仪”和“用户代表”感知障碍知识共享敏捷团队，和（ii）将调查结果纳入一个概念框架，包括这些看法，并提供管理敏捷开发的影响。为了解决这些目标，我们研究了以下的研究问题：如何在敏捷软件开发的关键利益相关者感知的障碍，有效的知识共享？

（The extant literature demonstrates several attempts to understand knowledge-sharing barriers or drivers within software teams. Focusing on software teams in general, Joshiet al. used 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 & D’Ambra described the dynamics through which outcomes, means and boundary interdependencies drive simultaneously cooperative and competitive behaviours 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 such as lightweight post-mortem styles and

sufficient documentation.The agile literature also acknowledges the driving impact of motivation-related factors, collaborative factors and ability-related factors While the general software development literature provides us with more solid theoretical foundations in this area, the agile literature is still fragmented with limited analyses based on and contributing to rigorous theoretical perspectives. 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 with its roots in sociology and offered insights for understanding behaviours and attitudes of individuals at work. The theory asserts that behaviour 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 behaviours attached to it. The focus of this theory on linking roles to beliefs and behaviours 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 behaviours. Specifically, building on previous research on key roles and their interactions, 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 the following: (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?）

研究方法：

比较因果图

比较因果映射是认知映射的一个变种，受访者通过访谈会话解释他们的因果断言的现象。它演示了模式的概念和因果信念，嵌入在不同群体的明确陈述。CCM的主要目的是和适合不同类型组织演员和精确的认知相似性或差异在这些演员之间的比较分析。此外，它最大限度的服务性和令人回味的利益与中小样本的定性研究的一个焦点。CCM已用于研究在个人或团体的钱德拉和发挥，2010知觉的相似性和差异性，因此本文的研究方法。

采用CCM对知识共享障碍感知不同角色的异同，我们认为该阶段的CCM：（i）数据预处理和数据收集阶段（参与者的选择，面试的设计，对称的数据采集和后续采访），和（ii）和数据收集分析阶段（建立编码指南和高阶类别，确定的因果关系，绘制地图的基础上体现主题，地图和寻求反馈专家分析）。我们的每一个阶段的适应是在下一个讨论中阐述的。

（Comparative causal mapping

Comparative causal mapping is a variant of cognitive mapping where respondents explain their causal assertions about a phenomenon through interview sessions . 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. Moreover, it best serves explorative and evocative interests with a focus on small or medium sample qualitative studies. CCM has been used to investigate perceptual similarities and differences across individuals or groups Chandra & Loosemore, 2010, 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 : (i) predata and data collection stages(participant selection, interview design, symmetric data collection and follow-up interview sessions), and (ii) analysis and post-data collection stage (establishing coding guidelines andhigher-order categories, identifying causal relationships, drawing maps based on reflected themes, analysis of maps and seeking feedback from experts). Our adaptation of each of these stages is elaborated in the next discussions.

）

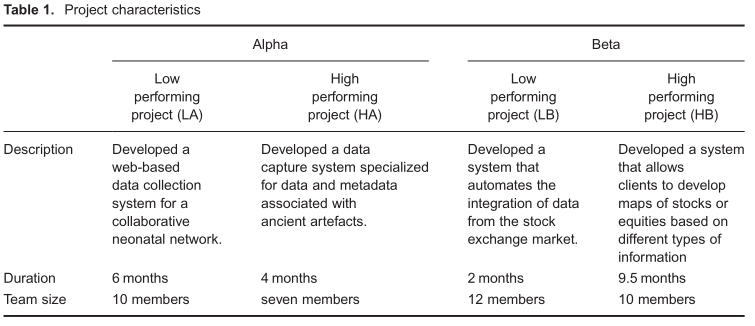
数据收集

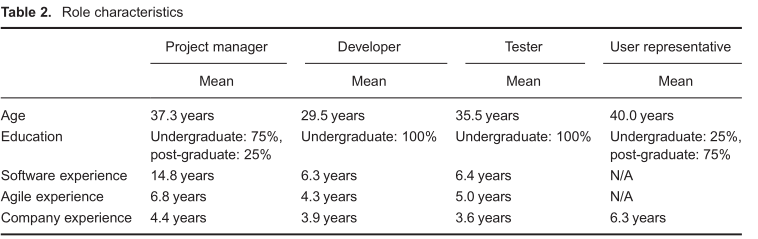
我们应用理论复制逻辑，以增加信心和对我们的研究结果的鲁棒性。更具体地说，我们使用了以下内容：（一）复制策略，包括一个不同的角度对知识共享的障碍，通过研究两个公司和选择参与者代表不同的角色，和（二）极性采样，以确保在选定的项目的差异，并添加相关的项目成果的见解，通过选择两个完成的项目在每个公司，一个被认为是高性能和一个被认为是低表现。

我们选择了两个澳大利亚的软件开发公司，为我们的研究体现的重要特征。澳大利亚软件产业已显示出越来越多的市场和技术的增长水平。α和β是中型，他们都遵循配置和软件项目的分布式敏捷方法。他们有一个很长的历史，专注于非商业软件开发与最近的商业项目的档案，展示他们的兴趣和建立的能力，扩大他们的活动的界限。我们收集的数据通过访谈会议，四个关键的角色，讨论在敏捷团队中有效的知识共享的障碍。我们与每个公司的开发经理密切合作，以确定和招聘八名受访者（两个项目经理，两个开发人员，两个测试人员和两个用户代表）从他们最近的敏捷项目。例如，持有项目经理的角色的个人的责任是确保客户的期望得到解决，在最终产品（两个公司使用术语“产品所有者”）。对于每一个公司，我们要求发展经理选择一个高性能和低执行项目的基础上四个指标：（一）利益相关者如何感知项目的表现，（二）如何处理团队的挑战，（三）如何将项目交付的期望和（四）社会关系如何经历。该项目在过去的3个月完成（在采访的时候），参与是自愿的，参与者被保密。

受访者被问及他们的看法，在考虑敏捷项目的知识共享的做法。敏捷的文献和现有的框架，在软件团队的知识共享的驱动程序指导的采访，但我们遵循了一个多层次的半结构化面试指南（附录），以唤起新的结构和联系，并创建多个场合获得人类的记忆。首先，我们收集了在指定的项目中的知识共享的背景信息，探索关键问题和术语，并提供“锚”的概念，为下一阶段的面试。对于这一点，我们问了开放的问题，如：“在什么方式是知识共享实践？第二，根据反应，我们要求探测的问题，引起进一步的信息的概念和相关的原因和影响的关系。第三、我们问的具体问题点到关键的推动者和障碍做最后的检查。这张给受访者关闭机会提醒我们任何项目，可能已经错过了或需要进一步解释。采访持续30min至1h，记录和转录。表1和2总结项目和受访者的特点。

在最初的访谈分析后，我们回到了阿尔法和测试版的后续访谈。我们的目标是解决含糊和验证解释的准确性和可信性，我们的解释。例如，有行情，方向性不明确，而讨论的上下文已指出因果关系。我们在起草具体的图形表示（因果图）



每个受访者和允许的评估是基于自己陈述的概念和联系的个人（共16个人的地图了）。我们使用半结构式访谈，以支持这一过程，通过问直接的问题，如：“当你说虚拟数据的测试是一个问题，你的意思是它抑制有效的知识共享？这导致了个人的因果图的小修改。

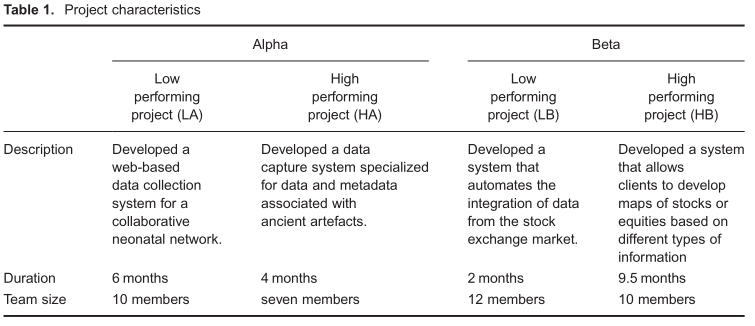
（Data collection

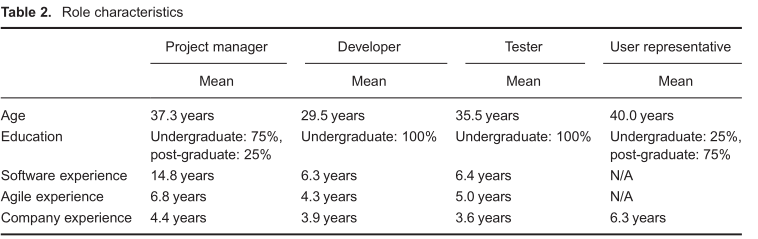
We applied theoretical replication logic to add confidence and robustness to our findings. More specifically, we used the following: (i) replication strategy to include a diverse range of perspectives on knowledge-sharing barriers by studying two companies and selecting participants to represent different roles, and (ii)polar sampling to secure variance across selected projects and add insights related to project outcomes 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 companiesthat exemplify important characteristics for our study. The Australian software industry has shown increasing levels of market and technological growth. 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 and two user representatives) from two of their recent agile projects. For example, the responsibility for individuals that held the role of a 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 3months (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 ) to evoke new constructs and linkages, and to create multiple occasions for access to human memory. 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 30min to 1h and were recorded and transcribed. Tables 1 and 2 summarize project and respondent characteristics.

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 . 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.）

数据分析

比较图

尽管因果映射的几个变种的广泛应用，CCM方法的讨论相对较少。我们采用了CCM使用cmap2软件由以下四个步骤：（一）劳克坎恩的方法创建和使用标准的词汇，（二）因果图的数据处理，（三）和（四）构建因果图因果图分析。我们利用不同的软件包，如编码和创建标准词汇NVivo，MS Excel处理因果矩阵和计算指标和MS PowerPoint创建视觉因果图。

在第一步中，“创建和使用标准词汇”，一个研究人员阅读所有的成绩单，分组经常提到的单词，并归纳产生了一个初步的代码列表。我们用一个标签来表示每一个代码，这些标签总结了被采访者使用的一些单词或短语的含义。然后，我们研究了所产生的编码方案的理论或逻辑上的相关性对现有的框架，对知识共享的驱动程序在软件团队，而不一定期待的诱发地图，以反映这些框架。另一个研究者阅读的材料来验证自己的表面效度评估编码方案的简约性和覆盖范围。一旦编码方案稳定下来，一位研究人员编码所有的转录和其他辅助阅读材料，以验证编码的有效性。在发生分歧，差异解决，通过讨论，以尽量减少研究人员的偏见。为了估计的编码过程中的可靠性，斯科特的pi计算使用16的八个访谈记录和34的编码类别。内容分析的一个启发式是需要约0.75或更多的可靠性系数，和史葛的皮我们的研究估计在一个可接受的水平0.86。

在第二步骤中，“因果图的处理数据”，我们系统地研究了所有的成绩单，以确定编码的陈述，通过使用关键字，如“所以”，“如果”和“因为”，意味着一个明确的因果关系的影响关系。我们记录了所有的因果陈述和联系的实际语言的受访者。然后，我们开发了一个列表的60个感知障碍（附录）通过分析确定的因果关系的陈述和联系，我们将这些障碍，莱维特的组织模式的人，结构，技术和任务，先前已被应用到分类软件开发风险。因此，我们确定了七个类别的障碍（见附录的详细信息）。例如，我们分类障碍相关的地理和时间上的差异，团队成员之间的差异（例如，团队成员之间的物理距离）为“团队多样性”的障碍（讨论的类别中提供的概念框架部分）。

在第三步骤中，“构建因果图”，我们代表了每个确定的“原因”和“效果”的概念在编码的语句（从前面的一步）与适当的高阶屏障结构（从以前的一步）。例如，采访声明：“我认为不同的文化[原因]是具有挑战性的知识共享[效果]。我们有敏捷的文化，可能与其他人不同，大学更官僚，不灵活，被编码为“敏捷与非敏捷文化（客户和发展公司）”（在第二步骤中的项目设置障碍分类）。当绘制因果图，因此，我们建立了“项目设置”和“有效的知识共享”之间的因果图之间的联系。另一个例子是，当受访者认为团队成员不熟悉（团队多样性的障碍）使他们更喜欢独立工作（独立工作，团队认知障碍倾向分类下），我们的团队之间建立联系的多样性”和“团队认知障碍。最后，我们结合四者相似的角色认知和可视化作为一个角色的具体因果关系图。这导致在四个因果图，代表了四个角色的看法。

在第四步，“因果图分析”，我们首先分析了地图的内容。内容分析抓住了一个沟通的主题和意义，一个个人（或组）认为是相关的。我们的后续访谈提供的成员检查，以验证解释的准确性，并检查每个单独的地图的可信性和可信性。下一步，我们的地图上和他们相比（例如，项目经理和开发人员的重点领域是什么）通过分析其结构伴随着面试陈述（伊甸et al.，1992；钱德拉&鲁斯摩尔，2010）。

结构分析的目的是检查地图内的项目之间的关系，以及它们是如何相互关联的。整合以前的建议，比较因果图，我们比较了地图在三个层次：地图，构建和结构之间的。为此，我们从定性访谈陈述得出推论（例如，多少次的受访者称项目沟通障碍及有效的知识共享之间的联系）和地图本身（例如，一些建设项目经理的地图）允许比较图。根据CCM文学、推理和报告结果需要艺术技巧，遵循因果关系的影响体现在初始和后续采访会议的地图信息。因此，除了使用分析措施（地图级，建设水平和结构之间的水平）为我们的比较分析提供了基础，我们也保持灵活的经验陈述和基本概念（见障碍强调部分或表5，例如）。总之，虽然分析涉及一些定量的数据，他们是定性的驱动，并旨在提供定性的见解。由于在定性研究中的习惯，我们使用和报告样本报价，把受访者的关键问题的生活。

地图层次分析比较了地图的综合性和密度。密度描述了构建地图的相互联系；它除以链接数在计算结构的数量结构图。高密度表示一个很好的理解的概念的受访者，而低密度意味着一个更简单，不太理解的概念。全面性是理解一个现象的深度和广度的一个指标，它是通过计算地图中的结构的数量来计算的。高综合反映多维的受访者的观点有关的概念，而较低的综合性是指局限在感知的概念从不同的角度。

地图层次分析比较了地图的综合性和密度。密度描述了构建地图的相互联系；它除以链接数在计算结构的数量结构图。高密度表示一个很好的理解的概念的受访者，而低密度意味着一个更简单，不太理解的概念。全面性是理解一个现象的深度和广度的一个指标，它是通过计算地图中的结构的数量来计算的。高综合反映多维的受访者的观点有关的概念，而较低的综合性是指局限在感知的概念从不同的角度。

构建层次分析比较地图的中心性结构的每个角色映射。中心性是一个指标，中央或重要的一个结构是地图，它是通过划分直接联系的数量，涉及到的结构的总数量的联系在地图上计算。在不同的地图的变量的理解的中心是很重要的，因为这项措施表明在不同的组的结构的感知的重要性的相似性和差异。我们研究和比较的中心性的障碍结构在每个地图了解和报告中心的每一个障碍的结构是为每个角色。

结构之间的层次分析检查的结构之间的可达性。可达性是两个结构之间的连接的总强度的一个指标，它被计算为一个结构上的另一个结构的直接和间接的影响的总和，我们检查在每个地图中的障碍结构之间的可达性了解和报告，每个角色强调指定的结构之间的关系（例如，产品之间的沟通障碍和有效的知识共享的关系，在开发人员地图）。方向性的联系可以用符号“+”或“-”，“+”表示在两因素呈正相关关系，而“-”表示成反比关系。

（Data analysis

Comparing maps

Despite the wide application of several variants of causal mapping, methodological discussions of CCM are relatively rare . We adapted Laukkanen’s approach for CCM using CMAP2 software by the 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 was calculated using eight 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, 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 ) 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 that has previously been applied to categorize software development risks. As a result, we identified seven categories of barriers (see Appendix 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 the previous step) with the appropriate higher-order barrier constructs (from the 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. Next, we made sense of the maps and compared them (e.g., what are the key focus areas of project managers vs. developers) by analyzing their structure accompanied by the interview statements (Eden et al., 1992; Chandra & Loosemore, 2010).

The objective of structural analysis is to examine the relationships among the items within a map and how they relate to each other. Integrating prior recommendations for comparing causal maps, 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 knowledgesharing) 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. Therefore, besides using analytical measures (map level, construct level and 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. As is customary in qualitative studies, we used and report sample quotes to bring the key concerns of interviewees into life.

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 . 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.

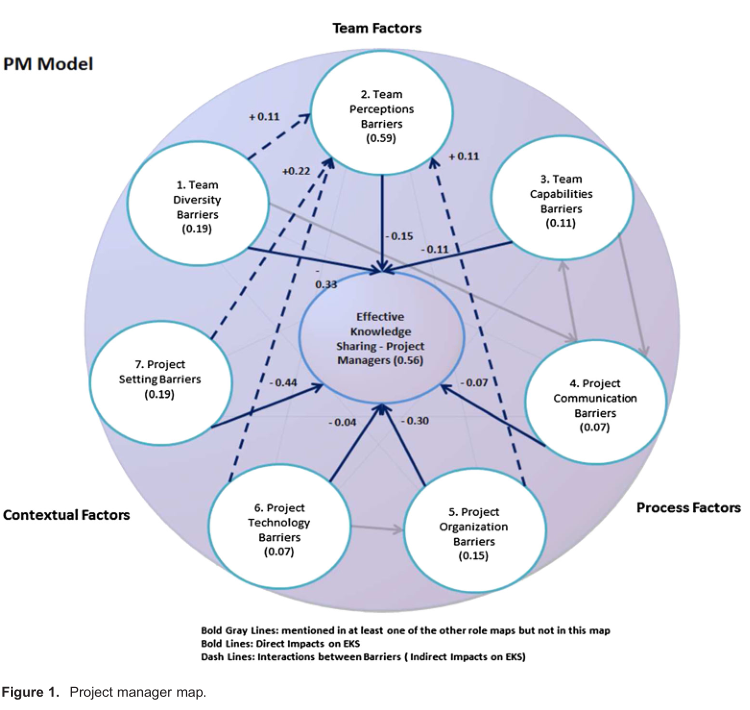
Construct-level analysis compared maps in terms of centrality of constructs for each role map. Centrality is an indicator of how central or important a construct is to the map; it is calculated by dividing the number of direct linkages involving the construct to the total number of linkages in the map. Understanding centrality of variables across different maps is important because 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.

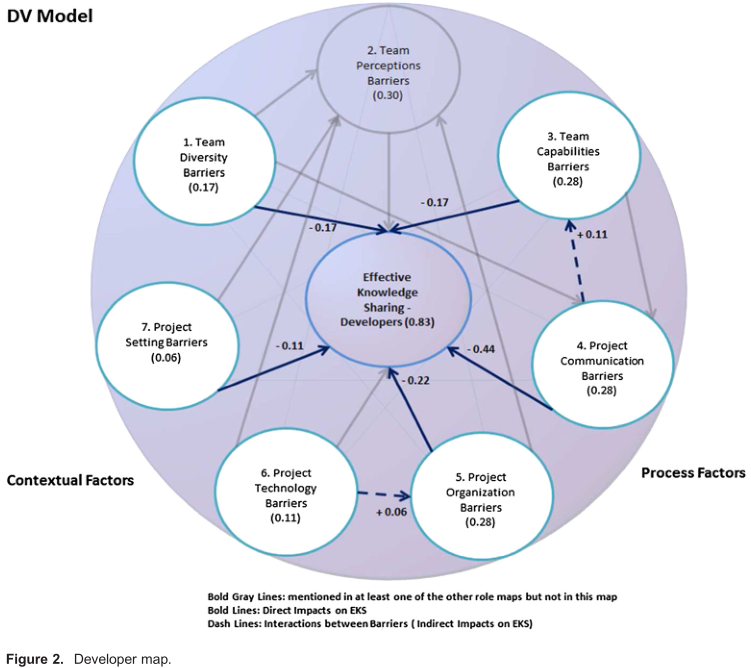
Between-constructs-level analysis examined reachability between constructs. Reachability is an indicator of the total strength of the connection between two constructs; it is calculated as the sum of the direct and indirect effects of one construct on another construct.We examined reachability between 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.

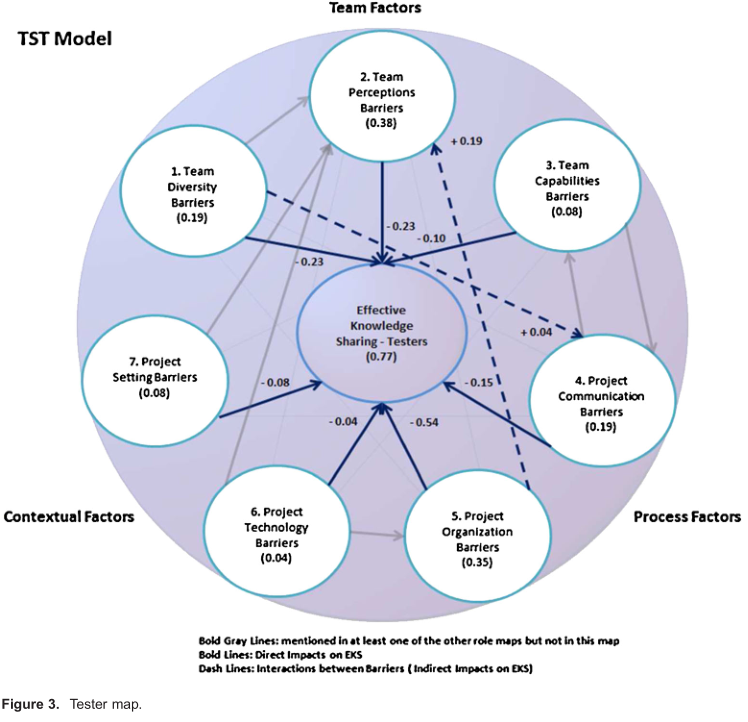
）

概念框架

其次，整合的障碍项目的结果（附录）和因果图（图1–4）

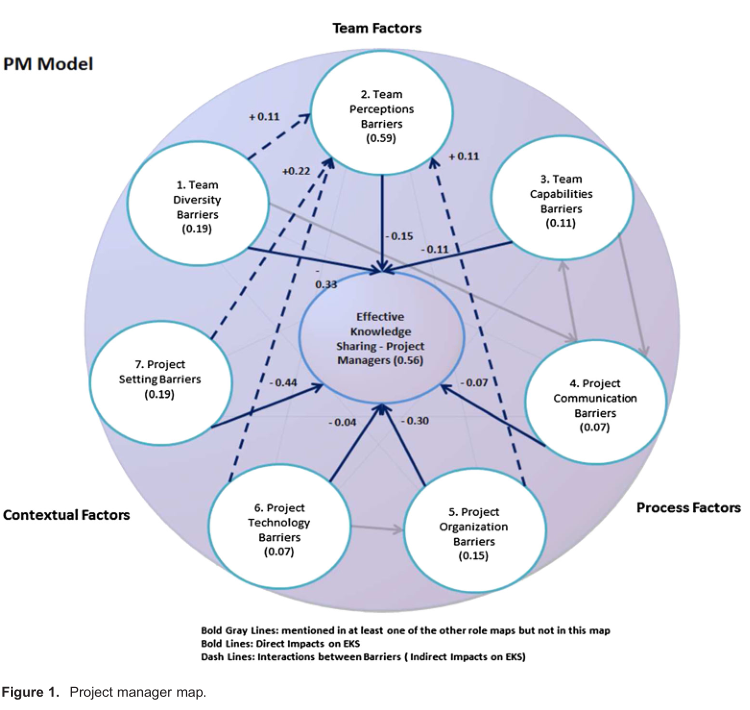
更为一般的实证结果，我们遵循三个步骤。首先，我们仔细地通过障碍结构和他们的相关的障碍项目（附录和数字4 - 1），删除多余的项目，合并类似和相关的项目，并采取了最初的步骤，从公司的具体术语。例如，“团队能力我们合并没有足够的IT人力资源相关的障碍下（客户公司）”的软件开发公司经验的缺乏（客户代表）和客户公司的软件开发经验不足。其次，我们出版物的补充和进一步发展的概念框架。作为一个，敏捷的文学影响的例子，我们增加了“基于康柏和摩根的开发团队的自我曝光不足的恐惧（2011）

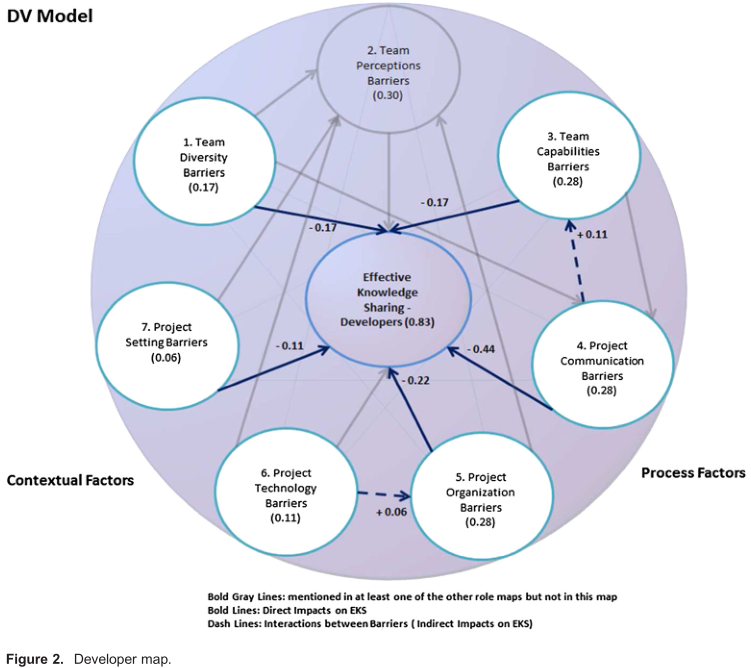
参考暴露弱点的恐惧通过高度密集的敏捷实践，如站立会议和现场客户。这种恐惧会使团队成员感到不安全和分享他们的想法。总共有六个项目被添加。第三，检查的应用程序和实用性的框架，我们提出了它的软件专业人员在阿尔法。我们逐步进行了四小时的评审会将收到的反馈每堂课最后的演讲前，接下来的评估。四届会议涉及（I）一个项目经理和开发人员，（二）一个项目经理，（三）一个测试仪和一个项目经理和（四）一个用户代表，分别。我们终止了评价的框架，在这一点上，因为最后的两个会话只导致了非常小的变化，参与者发现该框架的全面和建议使用它来管理敏捷项目。评估会议导致了修订的配方的10个障碍，以提高清晰度和提高了解。例如，通过开发团队忽视的非功能性需求修正为“利益相关者”忽略了非功能性需求。因此，我们提出了一个清单的37个知识共享障碍（附录C）和一般概念化的知识共享障碍（图5）。

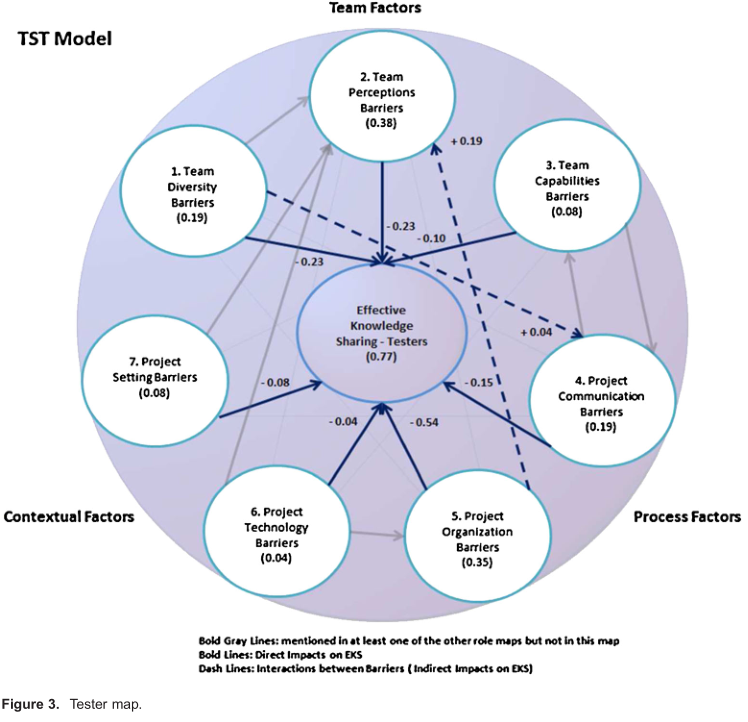


（Conceptual framework

Next, to integrate the findings from the barrier items (Appendix ) and causal maps(Figures 1–4) into more general empirical findings, we followed threesteps.

First, we carefully went through barrier constructs and their associated barrier items(Appendix and Figures 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 ITresources and software development experience in client company’.

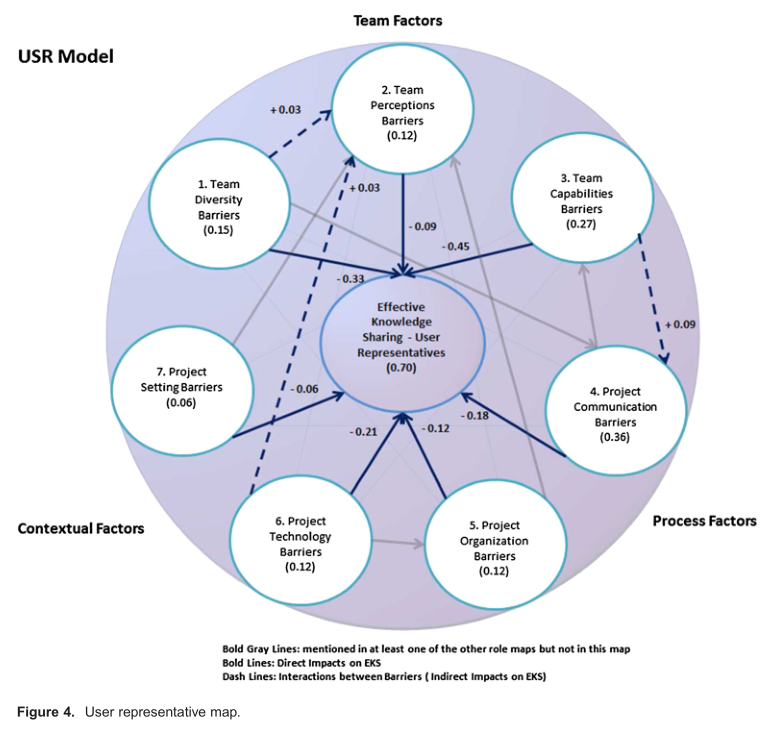
Second,we consideredcore publications onknowledgesharingintheagileliterature to complement and further develop the conceptual framework. As an .example of the impact of the agile literature, we added ‘fear of self-exposure deficiencies in development team’ based on Conboy and Morgan’s (2011) reference 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 honestopinions 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-h 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 10 barriers to increase clarity and improve understanding. For example, ‘Neglect of nonfunctional 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 C) and a general conceptualization of knowledge-sharing barriers(Figure 5).

）

结果

角色映射的比较

为了提供深入了解“如何在敏捷软件开发的关键利益相关者感知障碍，有效的知识共享”的研究问题，我们首先提出的因果图，从我们的分析项目经理，开发人员，测试人员和用户代表在α和β的访谈。数字1，2，3和4显示项目经理，开发人员，测试人员和用户代表的角色的因果图。

七个结构及其定义如下：

1团队的多样性障碍是指团队成员之间的概念，地理和时间上的差异，可能会阻碍有效的知识共享。

2团队感知障碍是指团队成员的态度和价值观，可能阻碍有效的知识共享。

3团队能力的障碍指的是团队成员之间的知识和技能相关的问题，可能会阻碍有效的知识共享。

4项目沟通障碍是指在项目内，可能阻碍有效的知识共享的通信相关问题。

5项目组织障碍是指可能阻碍有效的知识共享的项目的组织和行为的各个方面。

6项目设置障碍是指任务和上下文相关的问题，可能会阻碍有效的知识共享。

7项目技术壁垒指的是阻碍有效的知识共享的技术问题。

透明的灰色（衰落）结构和联系存在于整体的图，但不是在特定的角色映射。我们包括这些衰落的结构和联系，以提醒读者的整体图片，所建议的所有角色，并启用单一的图和汇总数据之间的视觉比较。例如，项目经理图（图1）表明，项目经理们提到所有七个知识共享障碍结构（综合图）。图的密度，它表示的结构的地图中的相互联系，为3.38（除以27计算（我们数提到了项目经理的采访，总的因果关系）8（多建在图））。在中心，在图1中最核心的结构是团队的看法和有效的知识共享结构（0.59，0.56），计算通过除以指定的结构（如在受访者的陈述中所述）的总数量的联系在图（16 / 27和15 / 27）的直接联系的数量计算。

在可达性方面，所有的障碍结构在图1的影响有效的知识共享的负面（大胆的链接）。可达性，这表明两个结构之间的连接的总强度，是最高的项目设置屏障的构建和有效的知识共享结构，这表明项目经理把一个很强的重点项目设置的知识共享的做法的负面影响。此外，一些结构相互影响（虚线链接）。例如，团队多样性的障碍导致了团队认知障碍（例如，不熟悉的团队成员，使他们更喜欢独立工作）。

要比较的图，在全面性，所有的图包括七个障碍结构，与异常，开发人员的地图不包括团队的看法建设。用户代表图的密度（密度= 4.13），相比其他地图（项目经理= 3.38，开发人员= 2.57和测试仪= 3.25）。关于核心团队的看法和措施，在项目经理的图设置的中央（0.59,0.19），相比其他图（开发商= 0，测试= 0.38，代表用户= 0.12；开发商= 0.06 = 0.08，测试人员和用户代表= 0.06）。这是预期的项目经理往往是在一个位置，观察团队的看法和态度，并报告他们。团队能力被认为是更核心的开发和用户代表性的图（0.28和0.27），相比其他图（项目经理= 0.11和测试人员= 0.08）。开发者和用户是移动发展的核心角色。因此，他们很自然地特别强调这一结构。我们还指出，项目组织被证明是更核心的测试图（0.35）相比其他地图（项目经理= 0.15，开发人员= 0.28和用户代表= 0.12）。测试人员的作用主要涉及与开发人员和项目经理，因此他们往往看到团队内部的组织问题，而开发人员正忙于开发和编码和项目经理都集中在与客户和促进开发商的谈判。最后，项目技术和项目通信被认为是相当多的中央在用户代表的图（0.12和0.36）相比其他角色。这可能是因为用户可能会受到发展公司和他们自己之间具有挑战性的沟通。开发人员的图是第二个强调项目沟通（0.28）。

在可达性措施方面，项目经理图中最强的联系是项目设置和有效的知识共享（0.44）之间的关系。这是可以理解的因为项目经理往往是大多数参与项目设置，如项目预算的障碍，组织政治和组织文化，这在敏捷开发会严重影响开发团队的能力，例如，在紧张的冲刺结束2提供示范。说明这种联系，一个项目经理说：

虽然当你有一个预算是小的，但是每一次的会议都算成是项目成本。所以，你必须平衡多少时间花在与人交谈和你花了多少时间做事情。

在开发人员图、测试人员图和用户代表图上的最强的可达性措施分别为：项目沟通障碍和有效的知识共享（0.44），项目组织障碍和有效的知识共享（0.54）和团队能力的障碍和有效的知识共享（0.45）。这些结果和应聘者陈述重新检查表明，最终用户（而不仅仅是客户密切沟通

重要的是，开发人员要更好地了解他们，并做好他们的工作。然而，这个问题可能会被忽略，在分布式敏捷项目，产品所有者主要负责与客户打交道。正如一位开发人员所说的：

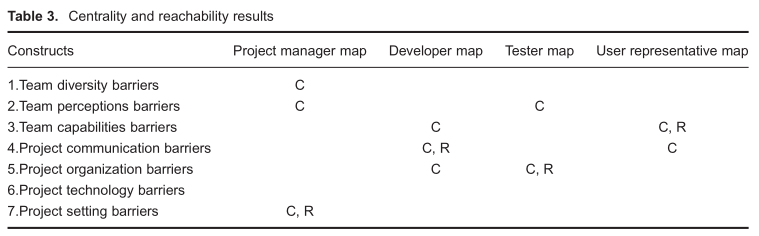
我在管理中的应用在生产主管。但我实际上没有太多的机会和那些实际使用一天一天的系统的人交谈的机会。也许这不是[项目经理]的情况。因为我知道他们确实去了[客户公司]了几次。

我们注意到，测试人员组织相关的问题放在周围如Scrum敏捷实践特别强调。具体而言，开发团队可能会试图坚持“个人和交互的敏捷价值的过程和工具”，而没有建立一个既定的结构和视觉审查会议。一个测试者说：

在拥挤的人群，我们去逛了一圈，你说你昨天做了什么，今天要做什么，如果你需要任何帮助。但他们往往成为一个地方，有人会说一点的东西，然后带出。因此，而更成为一个地方说的东西然后回去工作。

最后，用户代表往往归咎于开发团队（即，项目经理，开发人员和测试人员）缺乏了解用户的能力。这个问题在敏捷开发在软件开发的增量和沟通问题容易造成重大返工或错误的估计。正如一位用户代表说的：

我不太确定他们是如何[开发公司]培养他们的人收集知识，并把它翻译成计算机语言或系统。如果你问护士[最终用户]，他们不会给你任何直接的答案，因为他们没有任何软件开发的知识，他们可能会说是正确的，然后当你做测试时，他们会说：哦，这不是我们想要的。



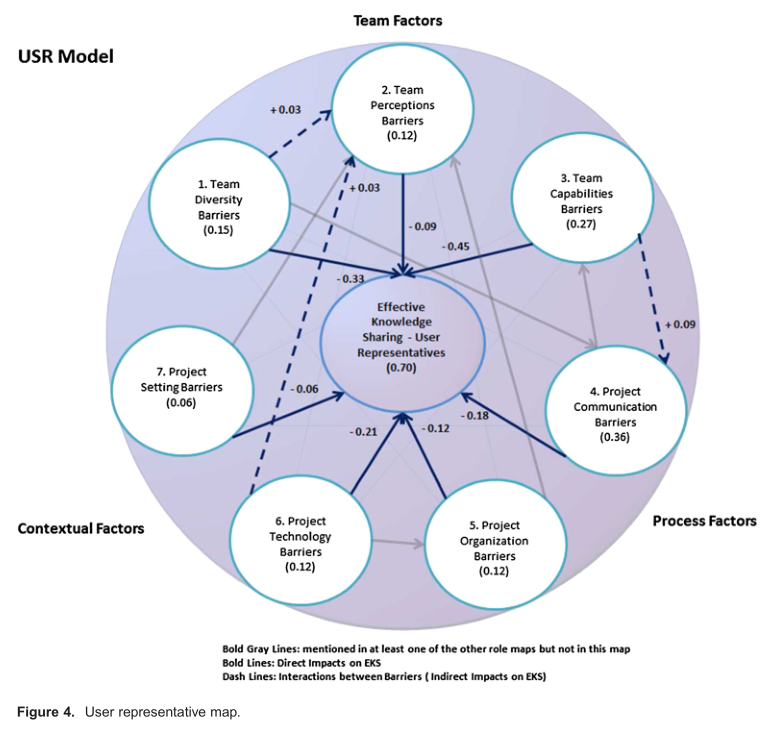
我们总结了在表3中的中心性和可达性的结果，显示出图的七个障碍结构中的每一个最强烈的重点。例如，项目设置是项目经理图中最核心的（我们在该小区中注意到一个中心）。此外，在项目经理图中的最强的链接是项目设置障碍和有效的知识共享之间的关系（我们注意到一个R在该单元格中的可达性）。

覆盖的中心性和可达性的结果，表3显示的项目设置、项目沟通、项目组织与团队能力的障碍是项目经理，开发人员，测试人员和用户代表各自的主要关注点不同。

（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. Figures 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 as follows:

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 among 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 grey (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).

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 to 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 and 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 and 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 and 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 and 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 because 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 2-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 re-examination 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:

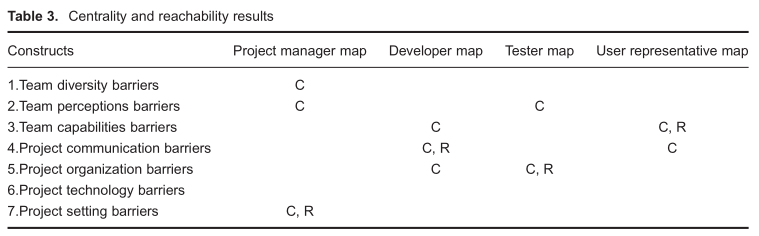
I am in charge of administrating the application after it is in production. But I haven’t actually had much chance to talk to people who are actually going to use the system on a day to day basis. Maybe that is not the case for the [Project manager]. Because I know they did go to the [client company]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

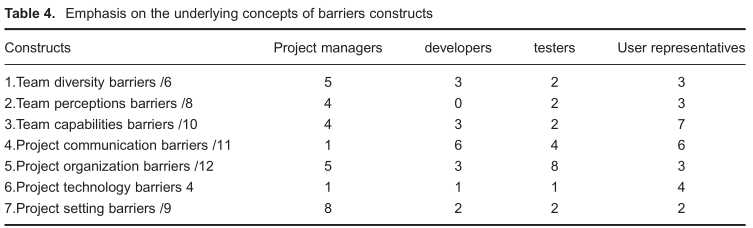
I am not quite sure how they [development company] train their people to gather knowledge and translate it into the computer language or the system. If you ask nurses [end users], they wouldn’t give you any straight answer … because they haven’t any knowledge of software development and they will probably 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).

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.）

强调跨越角色的障碍

要添加进一步的见解的比较分析，表4演示了如何强调每一个角色强调的七个障碍结构。

该表显示了如何在每一个结构的基本概念（见附录的结构和它们的基本概念）突出显示每一个角色，例如，第一行表4显示，在团队的多样性障碍，项目经理，开发人员，测试人员和用户代表六个概念，提出5，3，2和3个概念，分别。表4中的信息符合表3中的结果；每个角色建构最重要的障碍是在表3的大胆的项目相同。在接下来的讨论中，我们提供了如何构建一个详细的解释每一个角色实例化。

团队多样性的障碍

项目经理指六个五个经验观察到的概念在团队的多样性结构，而其他角色指的是较少的概念（3，2和3）。这是可以理解的，因为项目经理是在一个位置，观察团队的多样性在不同的群体和团队行为的后果。说明这一点，一个项目经理说：

我们在人们的知识方面有很大的差异。我们有一个知道[领域知识]超过五年的人。我们有那些从来没有工作过的人[这一领域]。

团队认知障碍

项目经理和测试人员有非常不同的看法相比，用户代表。他们提高团队相关的挑战（例如，低水平的重点项目），在一种情况下，他们也归咎于客户的态度（例如，不适当的假设由客户端）。正如一个项目经理所说的：

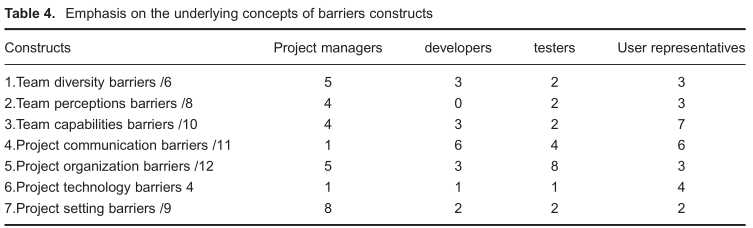
他们有一定的期望，他们可以得到的非常小的预算……它原来他们[客户代表]没有告诉我们这些……因为我们是灵活和灵活的，他们认为我们只是要这样做。

然而，用户代表点的项目技术和团队的看法之间的关系，解释不恰当的方法（敏捷方法）使开发团队不灵活的创新和变化。与之形成对比的是，开发团队避免了与敏捷方法论相关的挑战。一位用户代表说：

我发现，敏捷的格式限制创新，限制知识共享。如果他们[开发团队]没有一个写的故事，它似乎抑制他们的主动性，其实是积极主动的，做一些关于它的事情。我一直在会议上有明显的东西来了，我会说为什么这是没有做，为什么没有这个提出，有人说：我们没有一个故事。

（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 for constructs and their underlying concepts) are highlighted by each role.For example, the first row of Table 4 shows that out of six 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. In the next discussions, we provide a detailed explanation of how each role instantiated each construct.

Team diversity barriers

Project managers refer to five out of six empirically observed concepts within the team diversity construct whereas other roles refer to fewer concepts (3, 2 and 3). This is understandable because project managers are in a position to observe team diversity across different groups and its consequences on team behaviours. 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].

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.）

团队能力障碍

这种结构包括技术和社会团队的能力。在不同角色的关注近距离的观察表明，开发团队专注于技术壁垒（例如，不熟悉的语言）。正如一位开发人员所说的：

它是红宝石和轨道应用。我以前没有用过。嗯，我可能是主要的障碍，因为我不知道语言。

然而，用户代表强调了社会能力的重要性，为实现有效的知识共享。

[开发公司]需要了解[我们的业务]是什么，和什么人做[这个业务]看起来像。然后他们将能够与我们建立更有效的沟通。

此外，没有太多的一致性如何，项目经理，开发人员和测试人员与团队能力相关的实例化的障碍。测试人员的影响，往往是由项目经理提供的故事，他们与开发商的故事，所以他们提出的问题，开发和项目经理的能力。相反，项目经理对用户代表的能力是至关重要的（例如，缺乏软件开发公司的经验）。

（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).）

项目沟通的障碍

在项目沟通障碍的构建中的11个概念，项目经理，开发人员，测试人员和用户代表提出了1，6，4和6个概念，不同的角色，指的是不同的概念。开发人员将用户的代表（例如，与最终用户的功能不适当的检查），而用户代表批评的发展团队，他们的通信相关的限制（例如，被动与客户端的接触）。开发人员和测试人员似乎愿意有更多的面对面的沟通与用户代表，而项目经理认为这是更好的，如果他们负责与他们的客户。一个项目经理说：

我们有一个非常小的一组人，他们了解客户想要什么，可以创建一个共同的愿景，而不是每个人对如何工作都有自己的意见。

毫不奇怪的是，项目经理并没有提出许多关注（1个概念）关于项目沟通障碍。

（Project communication barriers

Out of the 11 concepts within the project communication barriers’ construct, project managers, developers, testers and user representatives 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.）

项目组织的障碍

测试人员参考12的八个经验观察到的概念在这个结构中，这是更多的其他角色（5，3和3）。用户代表不强调这一类，这是可以理解的，因为开发团队主要参与项目活动的组织。考虑到开发团队、开发人员、测试人员和项目管理人员不涉及相同的问题。测试人员主要是指项目组织的障碍，可能是因为他们是高度影响的开发人员（测试的故事）和项目管理人员（分配资源不足），因此往往强调团队内部的组织挑战。然而，开发人员经常忙于编码，项目经理主要集中在与用户代表进行谈判，并有利于开发人员：

我们在一个高峰，五个开发人员到一个测试。所以你在任何时候都会有大量的工作。所以你可以有一个基本的了解，但不是一个深刻的理解。

（Project organization barriers

Testers refer to eight 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 because 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.）

项目技术障碍

用户代表非常强调在这个结构中的所有概念（四个四），而其他的角色都关注一个概念。用户代表广泛质疑所使用的技术，而开发团队没有。开发团队提高了对合作技术的关注。正如一个项目经理所说的：

我们没有明确的定义的模板或方法来使用合流。它没有一套伟大的协作工具，至少是我们使用的版本。

只有一个开发商提出的敏捷方法相关的挑战，这个开发商只有对工作了7月敏捷项目。

（Project technology barriers

User representatives placed great emphasis on all the concepts within this construct (four out of four), whereas other roles were concerned about one concept. User representatives questioned the used 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 7months.）

项目设置障碍

这一障碍包括九个概念，项目经理指出，其中八个，而其他的角色只提到了两个概念的九。因此，项目经理非常关注项目设置对敏捷开发带来的挑战。这是可以理解的，因为他们的角色要求他们处理项目设置问题。例如，与其他角色相比，项目经理在一个非常好的位置，看看如何遗留系统，文化问题，组织政治和预算挑战的影响发展的做法。正如一个项目经理所说的：

他们[客户公司]有一个小的数据库，是由不是团队的一部分的人开发的。因此，他们对该项目的某些细节和要求的理解是不明确的，甚至是误导的。

（Project setting barriers

This barrier included nine concepts, and project managers pointed to eight of them, whereas other roles referred to only two concepts out of nine. 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:

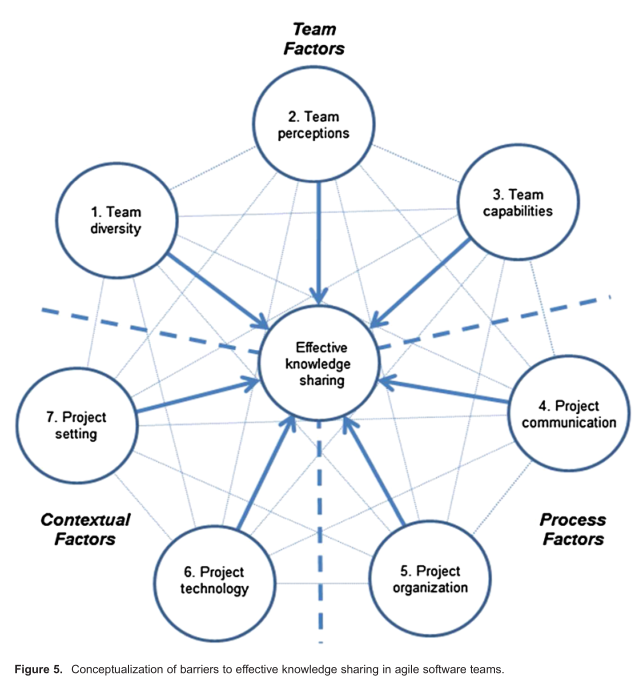
They [client company] had a small database that was developed by someone who was not part of the team. So, their understanding of certain details and requirements of the project was unclear and even misleading.）

障碍的概念框架

我们的最终框架由一个列表的37个知识共享障碍（附录C）和知识共享障碍的一般概念（图5）。

专注于提供一个概念化的知识共享障碍，图5揭示了三种类型的障碍（团队因素，过程因素和环境因素）和七个特定的障碍（团队的多样性，团队的看法，团队能力，项目沟通，项目组织，项目设置和项目技术）的基础上的37个障碍项目在附录C（如详细的数据分析部分）。

通过提供知识共享障碍敏捷团队中的一般概念，图5提供了肥沃的土壤，为今后的研究，特别着重于建立之间的屏障的相互关系的理论与实证模型的构建。类似于莱维特的模型描述了人的四个组成部分之间的所有可能的相互作用的结构、任务和技术（莱汀恩纽曼，2008），我们发现之间的联系，所有的屏障结构细的线条突出的可能关系，应该进行进一步的调查和模型构建。

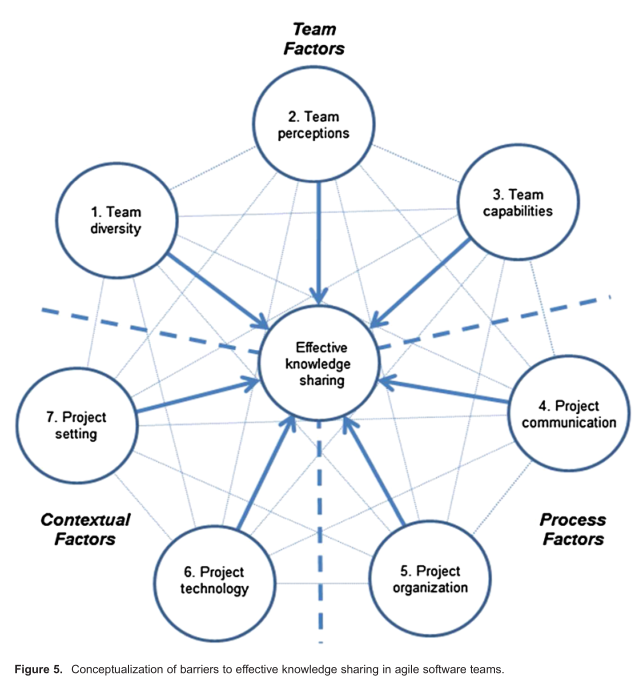


（Conceptual framework of barriers

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

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 and project technology) based on the 37 barrier items in Appendix C (as detailed in the Data Analysis section).

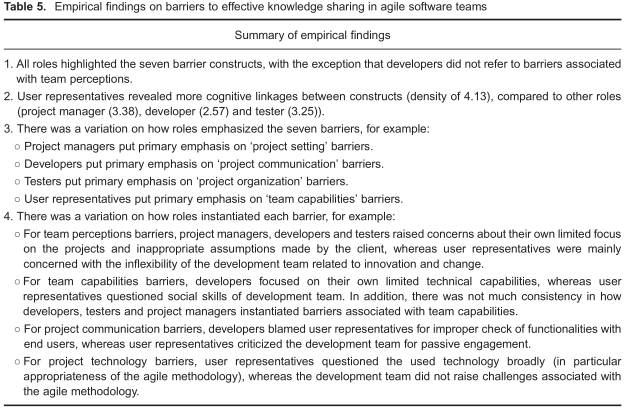
By providing a general conceptualization of knowledge-sharing barriers in agile teams, Figure 5 provides a fertile ground for future research that specifically focuses on establishing theoretical and empirical models of the interrelations between barrier constructs. Similar to how Leavitt’s model depicts 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.

）

讨论

对理论的贡献

本研究的目的是探索的相似性和差异的关键演员感知障碍，有效的知识共享在敏捷开发。制定我们的研究问题，“敏捷软件开发的关键利益相关者如何看待有效的知识共享的障碍？”'，我们首先要建立在组织中的角色理论和以前的系统开发的文献以四个关键角色的“项目经理”进行经常性的行为，'开发'，'测试'和'用户代表。知情的文献知识共享的一般软件团队的敏捷团队而言，我们使用CCM提供如何变成四个角色认为有效的知识共享障碍定性的见解。CCM使我们（一）唤起知识共享障碍的实证和广义的结构，（二）确定的论点，从面试陈述认知图的复杂链（图1–4）和（三）揭示认知的异同，如何观察团队角色认知的知识共享障碍。CCM的应用增加了丰富性和结构的研究，我们贡献了三个理论。



首先，我们已经提出了经验接地模型的不同观点在敏捷团队中的四个关键角色的知识共享障碍。这些研究结果代表的概括从理论（组织角色理论和模型的知识共享）和描述（访谈陈述）的描述。在表5中总结，障碍是，尽管有一定的相似性，不同的角色感知不同。具体而言，在结果部分中详细讨论，我们发现了显着的变化，如何角色强调的七个障碍（表第三中的5个点）。此外，不同的角色提到不同的障碍结构的实例（表第四中的5个点）。因果模型（图1–4）和总结在表5中支持先前的研究认为，角色模式的行动和相互作用，影响事件的进程，因此，他们可能会支持继续努力阐明和促进作用的软件开发环境。这些结果同意链接不同的角色执行任务的研究。例如，它是可以理解的，用户代表认为，开发商缺乏知识，他们的领域知识共享的障碍，而开发人员，反过来，可能认为在客户公司内部的政治挑战，反映在信息技术代表的频繁变化，作为主要障碍。此外，我们的研究结果揭示，敏捷环境独特的重要的感知差异，敏捷的价值观和原则（例如，欢迎变化和基于高质量的软件开发测试过程）可能会加强沟通相关的困难和矛盾。例如，用户代表解释开发商的僵化欢迎变化作为屏障，以创新和有效的知识共享，而开发商认为用户有不恰当的期望如何灵活的开发商可以而且应该，从而抑制有效讨论前面。在另一个例子中，虽然开发团队试图坚持“基于高质量的软件开发”的敏捷原则，他们往往忽视了教练，指导和自愿贡献在绩效评估的重要性。虽然这种感性的差异是不可避免的，认识他们如何表现在敏捷团队中的关键角色，揭示了新的见解有限的信息系统（是）的研究，侧重于组织角色理论和角色之间的互动。

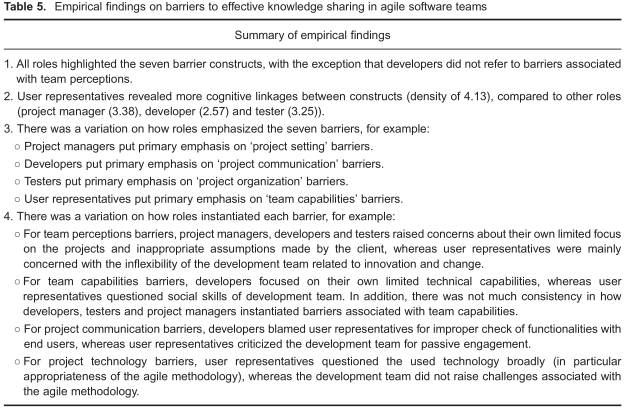
第二，我们借鉴了现有的知识共享和访谈报表模型，开发一个概念框架的知识共享障碍在敏捷团队（图5和附录C）。后李和维尔（2003），该框架是从理论和描述的理论概括，从而促进了移动敏捷软件开发作为一门学科。这可以说是一个更全面的和系统的概念化的知识共享障碍相比，以前被发表在敏捷文学和一般的软件文献。通过提出一个全面的图片的知识共享障碍（图5，附录C），我们的研究涵盖了一些一般性的软件风险项目和在敏捷实践中的经典错误。例如，障碍，如缺乏动机，不切实际的，估计和预期和不足的规划，也被提到在快速发展的经典错误。然而，我们的框架是不同的，在一些方式。首先，它显然是专注于知识共享的障碍，而不是一般的性能结果。其次，它探究如敏捷开发不是在类似的框架，提到了一些独特的挑战，通过指定的障碍，如缺乏目的和Scrum会议的合适的组织、决策发展没有咨询客户（由于紧张激烈的冲刺计划）”和“缺乏对客户的前面灵活的时间要求通信。此外，框架合并同类项，去掉多余的物品，并已采取措施，从公司特定的语言和术语（例如，合并“省略必要的任务从估计和过于乐观的计划”）。因此，在图5和附录C的框架打开定性和定量研究的知识共享屏障结构之间的动态相互作用，一行的调查，并没有得到足够的解决在以前的研究中。

最后，我们使用极性采样（Eisenhardt & Graebner，2007）的高性能、低执行项目安全的方差在选定的项目和增加的项目成果相关的见解。比较两个高绩效的项目之间的壁垒（HA，HB）和两个低表演项目（LA，LB），我们观察到一个总的8、15、30和20的知识共享项目HA，HB的障碍项目，La和Lb。因此，高性能的项目经验较少的知识共享的障碍比低绩效。认识知识共享障碍和绩效成果之间的关系（高性能项目经历了更少的知识共享障碍比低执行），未来的研究可能集中在如何高性能和低执行项目不同的方式，他们解决知识共享的障碍，如果有任何使用的分辨率技术和性能结果模式之间的关系。

（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 and previous systems development literature 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 generaland agile teams in particular, we used 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 (Figures 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 . 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 (third point in Table 5). In addition, different roles referred to different instances of barriers constructs (fourth 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 and as such, they may support continued efforts to explicate and promote roles in software development contexts. Some of these results concur with the previous studies that link different roles to tasks performed. 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 information technology 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 highquality software developed) may reinforce communication-related difficulties and inconsistencies. 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 . While such perceptual differences are inevitable, awareness of how they manifest across key roles in agile teams reveals new insights into the limited information systems (IS) research that focuses on organizational role theory and interaction between roles.

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 C). 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. 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 and the general software literature. By presenting a comprehensive picture of knowledge-sharing barriers (Figure 5, Appendix C), our study covers some general software risk items and classic mistakes in agile practices. 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. 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, 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 C 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.

Third, we add to the promising literature on CCM . 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, 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 used polar sampling 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 used resolution techniques and performance outcomes.）

管理的影响

个人和环境的需求施加不同的角色的看法和行为。虽然这些差异是自然的，意识到这种变化是软件项目的成功的关键。软件组织应考虑教育他们的开发团队有一个更好的理解知识共享的障碍和不同的关键利益相关者的看法（图1，5，附录C）。为了支持集体的专注，项目经理被鼓励参与团队成员在分析的障碍，在图5和37个确定的障碍模型。这样的集体努力可能会帮助团队成员共享心智模型，主动解决冲突，培养团队成员的合作。此外，管理者应该了解不同的方式障碍可能成为明显的（障碍强调跨角色部分）。例如，支持以前的文献软件团队的特点，我们发现在不同角色的团队能力的障碍的表现固有的差异。开发人员往往专注于技术方面的团队能力，而用户的代表强调了社会能力的重要性。考虑到这些因素，管理人员可能会重新招聘，培训和人力资源管理的敏捷团队。

要解决知识共享障碍（附录C），方法需要考虑的背景下，他们在同一家公司随着时间的推移，由于因素，如目前的发展团队的特点，客户的特点和项目的复杂性。敏捷的文献指出了一些方法来克服知识共享障碍，例如，视频发生器和王（2009）、识别等做法的回顾和对编程以及有利的影响的探索学习新的领域并与团队分享。知识管理文献是指一些方法，如“社区论坛”和“使用空间”，以促进知识共享。此外，软件开发的文献提供了提高知识共享和克服障碍，建议如马蒂亚森和沃格里斯昂（2005）讨论网络和网络知识共享网络的方法–方法强调知识共享技术的使用，而网络的方法把重点放在建立在那些参与软件开发的信任与合作。在制定任何方法之前，管理者应该意识到可能的抑制作用。康博伊和摩根（2011）显示，例如，每日站UPS可以减少团队成员的时间分享和讨论的想法，和案件的牙买加印度软件团队透露边界对象涉及定义的控制和权力的重新分布导致新兴的跨文化差异，又抑制了知识共享。对于知识的动态视图，知识共享实践的管理者应该考虑知识的固有特征，例如，知识是否主要是隐性的或显性的。知识管理的文献也区分系统为导向的战略重点集中在创建，存储，共享和使用的知识，通过信息技术（主要涉及显性知识），和以人为本的战略，强调通过社会和人际网络的对话（涉及隐性知识）。

（Managerial implications

Personal and environmental demands impose different role perceptions and behaviours. 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 C). To support collective mindfulness , 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, 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 C), 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, 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. The knowledge management literature refers to several approaches such as ‘community forums’ and ‘use of space’ to facilitate knowledge sharing . Moreover, the software development literature offers suggestions for improving knowledge sharing and overcoming barriers, 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. With respect to the dynamic views of knowledge , managers of knowledge-sharing practices should consider the inherent characteristics of knowledge, e.g., whether knowledge is mainly tacit or explicit . 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).）

局限性和未来发展方向

我们采用了定性的方法CCM来一点点了解知觉差异和知识共享行为的敏捷团队的领域。虽然CCM使我们认知图，可以作为进一步的定性或定量的探究指导，该方法是基于在一个有限的样本大小。此外，我们的研究是在澳大利亚的两个中型组织进行。我们建议进行大规模的实证研究，以验证、修改或扩展所提出的因果图。这样做，特别是大样本，我们鼓励使用可用的软件比较因果图如cmap3或Cognizer。研究人员可能会调查方法和干预措施，导致不同的角色有更多富有成果的对话，并可能分别映射到所确定的障碍、过程和产品相关的不确定性。

（Limitations and future directions

We used a qualitative methodology, CCM, in a little understood domain: perceptual differences to knowledge-sharing behaviours 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. 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.）