Information & Management 52 (2015) 82–97

Contents lists available at ScienceDirect

Information & Management

journal homepage: www.elsevier.com/locate/im

What drives knowledge sharing in software development teams: A literature review and classification framework

Shahla Ghobadi *

Australian School of Business, University of New South Wales, Sydney, NSW 2052, Australia

ARTICLE INFO

Article history:
Received 15 September 2013
Received in revised form 7 October 2014
Accepted 12 October 2014
Available online 22 October 2014

Keywords:
Software development
Software teams
Information system development
Knowledge sharing
Knowledge transfer
Literature review

ABSTRACT

Although scholars have long studied knowledge sharing drivers within software development teams, our knowledge remains fragmented by the divergent efforts that are based on and contribute to theoretical perspectives. This study provides a review of the extant literature (1993–2011) on knowledge sharing drivers in software teams and establishes a classification framework using an organizational change perspective. A synthesis of the literature uncovers diverse themes and gaps in the existing body of knowledge, suggests several paths for advancing theory on knowledge sharing in software development contexts, and discusses implications for practitioners concerned with knowledge sharing in software development.

1. Introduction

Software development is a collaborative and knowledge-intensive process that requires the blending and interweaving of diverse knowledge dispersed across domains of specialization [72,81]. The unique and inherent characteristics of software development signify the importance of effective knowledge sharing, *referring to the exchange of task-related information, ideas, know-hows, and feedback regarding software products and processes* [19], in exploiting available resources, addressing perceived challenges, and exploring emerging opportunities in software development and design [23,20,85]. For example, software, as a product, continuously emerges from intensive and iterative development and quality assurance cycles that require rapid reflections and frequent introspections across team members who represent different specializations, are often distributed, and may have opposing professional priorities [73,29,100,16]. Furthermore, from self-organizing open source communities [87] to distributed software development, which is rapidly becoming a norm in software companies, effective knowledge sharing is necessary to allow team members to discuss critical aspects of projects and overcome the cultural and social challenges of coordinating work across distributed spaces [17].

Nonetheless, several challenges may complicate knowledge sharing in software development teams [84,14]. For example, managing the diverse social identities and cross-functionality of team members [28,15], overcoming coordination challenges across distributed sites [17], creating homogeneous teams with a shared understanding [67] and motivating stakeholders to share embedded knowledge with development teams [18] can be challenging. Although technological solutions such as component-based development have been designed to facilitate rapid development and reduce the need for communication [50], research suggests that they have even exacerbated the need for knowledge sharing; for example, achieving ideal levels of component reuse requires developers to effectively share their knowledge of different components that they have developed and used [50].

Accordingly, researchers have studied knowledge sharing drivers in software teams (*factors that drive the exchange of task-related information, ideas, know-hows, and feedback regarding products and processes*) to propose effective ways of facilitating knowledge sharing within these contexts [100,15,61,30]. For example, Kotlarski and Oshri [49] emphasize the role of human-related issues, such as 'rapport' and 'transactive memory', in driving knowledge sharing within globally distributed software projects, Joshi et al. [45] employ a connectionist epistemological perspective and show the crucial role of 'source credibility' and 'extent of communication' in shaping knowledge transfer.

* Current address: University of New South Wales, Sydney, NSW 2052, Australia. Tel.: +61 2 9385 7130; fax: +61 2 9662 4061.
E-mail address: s.ghobadi@unsw.edu.au

http://dx.doi.org/10.1016/j.im.2014.10.008
0378-7206/

Exploring the role of transactive memory in distributed software teams, Oshri et al. [70] suggest that the 'standardization of templates and methodologies across remote sites', 'frequent teleconferencing sessions' and 'occasional short visits' support knowledge transfer. Pee et al. [73] use the lens of social interdependence theory to explain how goal, task, and reward interdependencies shape knowledge sharing between business and external IT consultant subgroups. Building on Pee et al.'s study, Ghobadi and D'Ambra [30] describe the dynamics through which interdependencies, including 'outcomes' (goals, rewards), 'means' (task-related), and 'boundary' (friendship, sense of identity, geographical closeness), interact and drive simultaneously cooperative and competitive behaviors, which in turn shape high-quality knowledge sharing in multi-party software teams.

Despite the valuable findings of previous studies, our understanding of this phenomenon is still emerging, and research is fragmented with limited integrated efforts that are based on and contributing to rich and rigorous theoretical perspectives. For example, whilst studies point to knowledge sharing drivers in software development teams [100,16,49,70], their focus is limited to examining a small and selected set of drivers; for instance, Chou and He [16] show the effect of 'open source software values and norms' on collaborative knowledge sharing in open source software teams, and Joshi et al. [45] highlight the importance of a 'source's credibility' and 'extent of communication' in shaping knowledge sharing practices. Furthermore, several research articles use terms such as 'participation' [2,5], 'requirement gathering' [34,33], 'activity engagement' [35], and 'sense giving' [90] to refer to the exchange of task-related information, ideas, know-hows, and feedback regarding software products and processes (knowledge sharing); for example, 'participation' and 'activity engagement' are used to refer to assisting others by answering their questions [5] and contributing to mailing lists in virtual open source development teams [35], and 'requirement gathering' has been reported as the process through which users share business and technical knowledge along with 'what they want from the product' with a development team [33]. In addition, researchers diverge in their underlying assumptions and research terminologies when studying knowledge sharing in software development teams, and consequently, a clear consensus in conceptualizations and findings has not been realized; e.g., the term 'knowledge transfer', which is primarily concerned with the internalization of the shared knowledge [45], is also used to refer to the simple sharing of knowledge [91,3,32].

Despite the importance of the subject, no research bridges multiple views and integrates existing findings into a comprehensive framework for researching and managing knowledge sharing drivers in software development teams. Although systematic literature reviews on the general aspects of knowledge management in software engineering exist [11], the lack of a focused review on knowledge sharing drivers within software teams and the paucity of efforts to integrate the existing fragmented knowledge complicate the prospect of understanding the current state of research and a continued discussion on the topic.

This study therefore aims to integrate existing findings, improve our understanding of the phenomenon of knowledge sharing in software development teams, and guide future research in this area. For this, a synthesis of the predominant literature with the following two research objectives is undertaken: (i) to identify patterns that emerge from previous research on knowledge sharing drivers in software development teams and (ii) to study the reported knowledge sharing drivers and integrate them into a framework that provides a rich picture of knowledge sharing drivers in software teams and facilitates studying knowledge sharing in software development contexts. To address the research objectives, the following steps are followed: (i) the existing literature is reviewed through the guidance of the following research question, *what drives knowledge sharing in software development teams?*, (ii) the contexts and terminologies referring to knowledge sharing and the employed research methodologies are extracted and analyzed, (iii) the reported knowledge sharing drivers are consolidated, classified and integrated into a classification framework, and (iv) the themes and gaps in the existing body of knowledge are highlighted, and avenues for future research are discussed. The remainder of the paper is structured as follows. Section 2 details the research methodology. Sections 3–5 elucidate the results of the literature review and synthesis and present the classification framework. Research and practical implications are discussed in Section 6, prior to outlining final remarks and research limitations in Section 7.

2. Research method

Fig. 1 demonstrates the three methodological phases along with their relevant steps. Based on the systematic mapping method [76,6], which focuses on categorizing the research phenomenon under investigation and visualizing findings into a structured framework, the methodology consists of three major phases, including: (i) planning the study, (ii) conducting the study, and (iii) reporting the review. Sections 3–5 explain each of the three phases, supplemented by a detailed discussion of the results and avenues for future research in Sections 6 and 7.

3. Phase 1: planning the study

The first and second steps in 'planning the study' involved 'identifying the need and rationale for the study' and 'formulating the research question'. The third step was the 'development of a review protocol' to guide the review; for this, an initial review protocol was written and then was revisited and improved by two

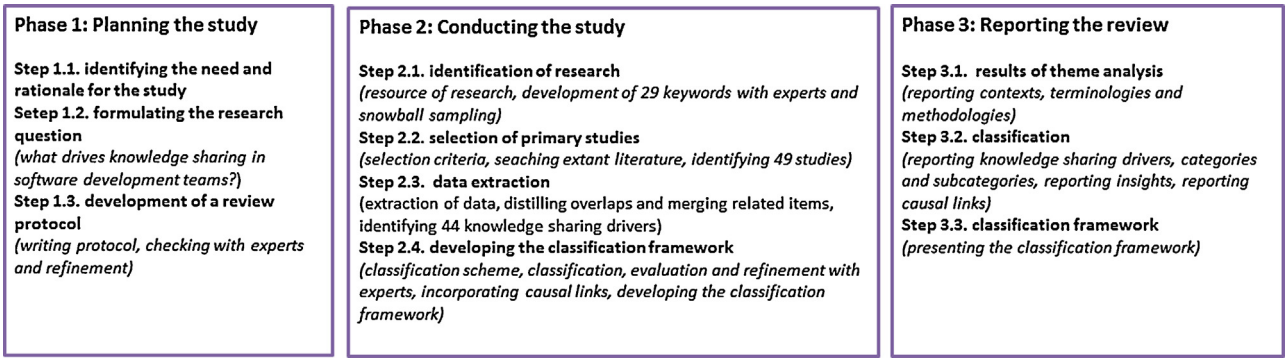

Fig. 1. Methodological phases.