Information and Software Technology xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Information and Software Technology

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

Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study

Sherlock A. Licorish *, Stephen G. MacDonell

Department of Information Science, University of Otago, PO Box 56, Dunedin 9054, New Zealand

ARTICLE INFO

Article history:
Received 30 March 2013
Received in revised form 13 January 2014
Accepted 11 February 2014
Available online xxxx

Keywords:
Core developers
Psycholinguistics
Content analysis
Attitudes
Knowledge sharing
Task performance

ABSTRACT

Context: Prior research has established that a few individuals generally dominate project communication and source code changes during software development. Moreover, this pattern has been found to exist irrespective of task assignments at project initiation.
Objective: While this phenomenon has been noted, prior research has not sought to understand these dominant individuals. Previous work considering the effect of team structures on team performance has found that core communicators are the gatekeepers of their teams' knowledge, and the performance of these members was correlated with their teams' success. Building on this work, we have employed a longitudinal approach to study the way core developers' attitudes, knowledge sharing behaviors and task performance change over the course of their project, based on the analysis of repository data.
Method: We first used social network analysis (SNA) and standard statistical analysis techniques to identify and select artifacts from ten different software development teams. These procedures were also used to select central practitioners among these teams. We then applied psycholinguistic analysis and directed content analysis (CA) techniques to interpret the content of these practitioners' messages. Finally, we inspected these core developers' activities as recorded in system change logs at various points in time during systems' development.
Results: Among our findings, we observe that core developers' attitudes and knowledge sharing behaviors were linked to their involvement in actual software development and the demands of their wider project teams. However, core developers appeared to naturally possess high levels of insightful characteristics, which became evident very early during teamwork.
Conclusions: Project performance would likely benefit from strategies aimed at surrounding core developers with other competent communicators. Core developers should also be supported by a wider team who are willing to ask questions and challenge their ideas. Finally, the availability of adequate communication channels would help with maintaining positive team climate, and this is likely to mitigate the negative effects of distance during distributed developments.

1. Introduction

Previous research has established that a few individuals in a team generally dominate project communication and source code changes during software development [1–4]. Evidence has also shown that, even in environments with fixed task assignments, specific individuals circumvent these pre-set arrangements to occupy the center of their teams' activities [5]. Such patterns have been studied previously in other disciplines [6,7], and early works investigating the effect of this phenomenon have shown that the existence of these centralized patterns involving core group members is a positive sign for team performance [8]. In similarly seminal work, Leavitt established that central individuals are vital to their teams' performance as they coordinate information flow. Central individuals are also *seen as* project leaders, whether or not they are the formal or nominal leaders [9], and groups with central coordinators experience higher levels of group organization and task performance (in terms of speed when completing tasks) [8].

While there is therefore strong interest in identifying patterns within software teams' communication and coordination practices, there has been comparatively little effort directed toward understanding why these patterns exist or how they emerge. Questions related to how core members share knowledge over their project,

* Corresponding author. Tel.: +64 3 479 8316; fax: +64 3 479 8311.
E-mail addresses: sherlock.licorish@otago.ac.nz (S.A. Licorish), stephen.macdonell@otago.ac.nz (S.G. MacDonell).

http://dx.doi.org/10.1016/j.infsof.2014.02.004
0950-5849/

the initial arrangements that lead to core members becoming hubs in their teams, and how the attitudes and traits these practitioners exhibit might be linked to their involvement in task changes, have not been answered. Such explorations could provide insights into the peculiarities of software team dynamics, inform appropriate team configurations, and enable the early identification of 'software gems' – exceptional practitioners in terms of both task and team performance.

In previous work we used psycholinguistics and content analysis techniques to study the roles and behaviors of core developers and uncovered that these practitioners worked across multiple roles, and were indeed crucial to their teams' organizational, intra-personal and inter-personal processes [10]. Additionally, we noted that although these individuals were highly task- and achievement-focused, they were also largely responsible for maintaining positive team atmosphere [10]. While we established that core developers exhibited significantly different attitudes and behaviors to their 'regular' counterparts during team work, this prior study took a static, single snapshot view of the software project teams considered. Additionally, our goal in that prior work was primarily to examine *if* core developers behaved differently to their less active counterparts.

In building on that work we now answer the questions noted above in this paper. In the current work we employ a multi-stage approach to study the ways in which core developers' attitudes and knowledge sharing behaviors change over time. In order to do so we examine actual software artifacts as against the more frequent use of surveys and interviews [11]. Our analysis was conducted in several steps, described in detail in Section 3. Briefly, in the first step repository data was mined and artifacts from multiple software development teams were explored using social network analysis (SNA) and standard statistical techniques. This enabled us to select our cases and to detect patterns around core developers, and so informed the design of the later steps involving deeper linguistic analysis techniques and directed content analysis (CA).

Our work makes several contributions. In terms of methodological contribution, we demonstrate the value of employing contextual analysis techniques to understand internal software team processes. We extend software engineering (SE) theory and explain how and why core developers become knowledge hubs, we provide understandings for the way core developers' attitudes and knowledge sharing behaviors are linked to their involvement in software tasks, and we outline several avenues for future research. We also provide recommendations for software project governance and show how the outcomes of our work have implications for team strategies. Finally, we provide suggestions for the enhancement of collaboration and process support tools.

In the next section (Section 2) we present our theoretical background and survey related works; this leads to our specific research questions. We then provide details of our research setting and measures in Section 3, introducing our techniques and procedures in this section. In Section 4 we present our results and we analyze and discuss our findings. Section 5 outlines the implications of our results, and in Section 6 we consider our study's limitations. Finally, in Section 7 we draw our conclusions.

2. The study of team communication

Previous work has established that the intricacies of team dynamics can be revealed by studying members' communication [12]. Research has also uncovered linkages between informal hierarchical communication structures and team performance for geographically distributed teams [13]. Furthermore, team communication has been linked to coordination efficiency [14] and the quality of software output [15]. Thus, studying the details in team communication can provide valuable insights into the human processes involved during software development, including the reasons for, and consequences of, communication and coordination actions.

Given this, software repositories and software history data have emerged as valuable sources of interaction and communication evidence [16]. Research findings drawn from works examining such sources are particularly valid if the data represents the primary means of interaction and so captures team processes during software development [10]. Accordingly, previous researchers have exploited process artifacts such as electronic messages, change request histories, bug logs and blogs to provide unique perspectives on the activities occurring during the software development process [3,17]. In particular, open source software (OSS) repositories and archives recording software developers' textual communication activities have increasingly provided researchers with opportunities to study software practitioners' behaviors [18,19].

For instance, Abreu and Premraj [20] analyzed the Eclipse mailing list and found that increases in communication intensity coincided with higher numbers of bug-introducing changes, and that developers communicated most frequently at release and integration time. Bird et al. [4] employed clustering algorithms to study CVS records and mailing lists and concluded that the more software development an individual does the more coordination and controlling activities they must undertake. These observations are supported by Cataldo et al. [3], whose SNA study found that central individuals contributed the most during software development. The Debian mailing list was used by Sowe et al. [21] to observe knowledge sharing among developers. These authors found that no specific individual dominated knowledge sharing activities in the Debian project.

Works such as those of Bacchelli et al. [22] and Antoniol et al. [23] have used rather more complex techniques to analyze email and bug description information. In linking email communications to changes in source code using regular expressions and other information retrieval approaches, Bacchelli et al. [22] found that the analysis approach using regular expressions in emails outperformed more complex probabilistic and vector space models. Through the use of decision trees, naïve Bayes classifiers and logistic regression, Antoniol et al. [23] were also able to classify bugs based on specific terms used in the textual descriptions of such tasks.

In reviewing these two streams of work, it is evident that some researchers have looked to infer the semantics of practitioners' dialogues from the text they communicated (e.g., [22,23]), while others have provided deductions based on communication frequency information [4,20]. While text analysis methods and their associated tools have been used previously to understand and predict some aspects of software development [24,25], only a few studies in this domain have considered examining software teams' internal behavioral processes as represented in their members' textual communications. This is in spite of the fact, as noted by Bacchelli et al. [22], that natural language analysis techniques have proved to be effective in generating understandings of software developers' attitudes when applied to their language processes.

For instance, in analyzing the communication of the developers involved in the Apache project, Rigby and Hassan [12] uncovered that once the two most active developers signaled their intentions to leave the project their communications became more negative and instructive, they spoke mostly in the future tense, and they communicated with less positive emotions, when compared to their earlier communications. In our own work examining three different IBM Rational Jazz project teams we also found slight variances in behaviors among those undertaking different forms of software task [25]. As noted in Section 1, as a first step to understanding the true role of active communicators, we used linguistic

Please cite this article in press as: S.A. Licorish, S.G. MacDonell, Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study, Inform. Softw. Technol. (2014), http://dx.doi.org/10.1016/j.infsof.2014.02.004