STUDY OF FACTORS INFLUENCING THE ADOPTION OF AGILE PROCESSES WHEN USING WIKIS

ALBERTO HEREDIA*, JAVIER GARCIA-GUZMAN† and ANTONIO AMESCUA-SECO‡
Department of Computer Science, Carlos III University of Madrid
Av. Universidad 30, 28911, Leganés, Madrid, Spain
*alberto.heredia@uc3m.es
†jgarcia@inf.uc3m.es
‡amescua@inf.uc3m.es

ALAN SERRANO
Information Systems and Computing, Brunel University
Uxbridge, UB8 3PH, United Kingdom
Alan.Edwin.Serrano-Rico@brunel.ac.uk

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The adoption of innovative Software Engineering (SE) processes by an organization implies that engineers have to learn new processes they might not be familiar with. Social software can support and enhance this adoption process, so research needs to focus on how the exchange of knowledge among software engineers using these tools can help to perform training more effectively. We propose a framework based on social software to support the collaborative learning, adoption and improvement of SE processes through the exchange of experiences among individuals. This article examines factors influencing the adoption of new SE processes and the quality of the experiences shared using the proposed framework in comparison with similar ones. Two case studies were carried out involving junior engineers in a training course on agile software development. Anonymous surveys collected data on the perceived quality of the experiences shared during the research, their usefulness, and the simplicity of the mechanisms provided to contribute experiences. Results show that the adoption of new SE processes is influenced by several factors such as the commitment of software engineers to collaborate in the adoption of the new process, the perceived level of usefulness of the tacit knowledge elicited during the adoption process, the diversity of the topics covered by the shared knowledge, the simplicity of the mechanisms to contribute new tacit knowledge, and the amount of learning achieved by software engineers.

Keywords: Process adoption; social software; tacit knowledge; collaborative learning; agile software development.

1. Introduction

Software engineers working in innovative companies must deal with situations in which they have to acquire new knowledge to learn and adopt —in an efficient manner— a process that is unknown for them a priori. It usually happens when a Software Engineering (SE) organization wants to introduce a new process or when a newcomer engineer joins a project team.
The knowledge and experience acquired in previous projects can be useful in these situations so that the learning based on that experience becomes a key element in the effective adoption of new processes and for improving both productivity and quality [1]. This kind of knowledge is however difficult to find, and when found its reuse is difficult to achieve in practice [2, 3]. The main reason is that experiences from previous projects are context-specific and confined to each individual; its tacit nature makes this type of knowledge hard to formalize and to communicate [4], so other software engineers that may perform similar tasks and come up against similar problems do not take advantage of these experiences and successful practices are not repeated [1].

Knowledge Management (KM) initiatives attempt at creating an environment that supports the handling of knowledge and experiences within an organization to improve personnel competences [2, 5], which ultimately leads to increased organizational effectiveness [6]. One of the fundamental structures in every KM system is the knowledge repository. It stores knowledge artifacts in such a manner that they can be retrieved and reused [7] in order to help SE organizations in their effort to enhance the quality of their software products by adopting new processes or improving the existing ones [8], as the quality of software is directly related to the quality of the process through which software is developed [9].

Social software, such as instant messaging, blog and wiki, can be used to support the adoption of innovative SE processes among software engineers within an organization because it allows people to learn easily from knowledge of others [10]. Among the different social software, wikis were considered suitable for our research because they provide a lightweight repository to store the explicit knowledge about the existing organizational processes and also allow their users to collaboratively enrich that knowledge by contributing new contents derived from experience that have the potential for reuse in future projects. This tacit knowledge that comes from experience —made explicit in the form of examples, discussions or lessons learned— provides the knowledge required for the effective application of the technical and methodological knowledge (i.e. SE processes) [1].

In consequence, we focus our study on how to establish mechanisms to enrich the existing organizational knowledge about processes by adding software engineers’ experiences to a repository, making it easier to learn and adopt new processes. More specifically, we first propose a framework to enhance collaborative learning of SE processes. Based on KM principles through the use of wikis, the framework is initially targeted at —but not restricted to— environments where collaborative learning is essential (e.g. junior software engineers).

The main contribution of this article is the study of factors influencing the adoption of innovative SE processes —such as agile— through the analysis of factors that affect the quality of the experiences shared using the proposed framework based on social software. Similar studies [11, 12] previously identified organizational factors which are critical for the success of organizational learning and software process improvement. Our research however focuses in depth on factors related to individuals who have to learn and adopt a
new process and factors related to the tacit knowledge that is generated during the learning of that process. This work will contribute to study how to perform training on new methods more effectively.

The remainder of this paper is organized as follows. The next section contains a brief background on KM and an overview of existing studies on the application of KM principles to support the adoption of new SE processes using social software. Then, section 3 proposes our framework to support the learning, adoption and improvement of new SE processes. Section 4 describes the case studies within the scope of this research. In section 5, data obtained during the research is pre-analyzed and the results allow the extraction of some factors that may influence the adoption of SE processes. The discussion of the results according to these factors is presented in section 6. Section 7 discusses the threats to the validity of the study presented in this paper. Finally, in Section 8 we present our conclusions of the research work.

2. Background

KM has a great impact on SE [1, 13, 14] and is considered to be an essential part of software engineers’ training [15]. In SE organizations, KM can be used to capture and share the knowledge and experience generated during the execution of organizational processes [6]; although every project is unique in some sense, similar experiences can help software engineers to better perform their activities, and managers to capture the domain knowledge that software engineers acquire during their work [1, 16].

The knowledge that SE organizations have to manage can be classified into explicit and tacit knowledge [4]. The first is formal and systematic, and can be expressed without ambiguities in words, data, numbers, and language. On the other hand, tacit knowledge is personal, context-specific, and hard to formalize and to communicate among people because it is derived from experience [17]. Both types of knowledge must be managed properly to make organizational knowledge available for software teams in order to learn, adopt and improve the processes in a SE company [14].

KM simplifies the process of sharing, distributing, creating, capturing, and understanding organizational knowledge to enhance value [14]. To do so, knowledge repositories store artifacts in such a manner that they can be retrieved and reused [7]. In software development, the reuse of these software artifacts leads to lower defect density than if they were built from scratch, which therefore helps to improve the quality of the system [8] and the organizational performance [6].

Among these repositories that can manage all the organizational knowledge in a SE company is the Process Asset Library (PAL). It is a knowledge repository used to store and make available all the process assets that are useful to those who define, implement, and manage software processes in the organization [18]. A PAL not only contains knowledge about how to perform the processes but also stores the lessons learned from previous projects. Transferring these lessons learned across current and future projects is a key factor to develop dynamic competitive capabilities [19]. To do this, experience from software engineers has to be gathered, disseminated and reused [1].
Social software and especially wikis can be used as a technological platform for the collaborative management of all this software processes knowledge [10]. A significant volume of SE research focuses on the usage of wikis to support activities such as requirements elicitation [20], architectural design [21], coding [22], technical documentation of software [23], sharing and reusing experiences [24], and so on [10, 25].

Although there are many studies on the use of wikis in SE, few of them use wikis as PALs for software engineers’ training. Wongboonsin and Limpiyakorn [26] propose a PAL using a wiki to store process assets using CMMI as improvement process model. Amescua et al. [27] developed a PAL using a wiki as a lightweight repository of knowledge about agile software development, providing a more specific process structure for the PAL creation and management. The authors also suggested that further research has to be done into the definition of mechanisms to feed back the tacit knowledge and the definition of mechanisms and technologies for the configuration management of the organizational knowledge.

In relation to the definition of mechanisms to feed back the tacit knowledge, Falbo et al. [28] described an initiative to capture informal knowledge by storing some information about lessons learned. Each lesson learned has to pass a knowledge filter to decide whether it will be available in the repository or not. The main drawbacks of this solution are that only the project manager can input a lesson learned in the informal knowledge base, and the proposed solution has not been validated.

He et al. [29] developed a framework to capture process experiences, knowledge artifacts and personal skills from users to store them in a repository and to make them reusable. Although this solution offers a Web portal, it does not support a clear collaborative framework and the authors did not provide results from its use.

On the other hand, Eclipse Process Framework Composer [30] and IRIS Process Author [31] are tools that allow publishing in a wiki the organizational knowledge stored in a repository, so users can provide feedback and suggest improvements by modifying the published version. However, these changes are not automatically propagated back into the repository, making it difficult for other users to learn from previous experiences.

More recently, Jung [32] proposes a semantic wiki-based KMS to improve task performance by collecting as many organizational resources as possible and maintaining semantic consistency of the system. The semantic annotations provide more informative data, resulting in better performance than traditional wiki-based KMS.

Finally, Andrade et al. [1] presented the architectural model of a system to gather, store, disseminate and reuse software testing experiences and whose design tries to avoid the weaknesses detected in that type of systems. This approach is based on the management of the lessons learned that software testing engineers gather from everyday experience.

As we have seen, software engineers can use social software to communicate, coordinate and learn from one another. Despite the apparent widespread use of social software within SE, there are few empirical studies about the different factors that may influence the adoption of innovative SE process when using these technologies and the
quality of the tacit knowledge shared through them. Dybä [33] extensively reviewed the literature to identify critical factors of quality management, organizational learning and software process improvement. Later, he extended the study and identified six organizational factors which are critical for success: business orientation, leadership involvement, employee participation, concern for measurement, exploitation of existing knowledge, and exploration of new knowledge [11]. On the other hand, Mangalaraj et al. [12] provided insights into individual, team, technological, task, and environmental factors that facilitate or hinder the organization-wide adoption and eventual acceptance of agile practices. But further research is needed to focus in depth on factors related to individuals who have to learn and adopt a new process and factors related to the tacit knowledge that is generated during the learning of that process.

3. Proposed Framework

Our main objective is to examine factors influencing the adoption of new SE processes that may lead to a more effective training. To do so, we first need to describe the KM-based framework we use to enhance collaborative learning and adoption of SE processes, named ITAKA (Interactive TAcit Knowledge Administration) [34]. The framework (Fig. 1) aims at managing both explicit knowledge about existing SE processes and tacit knowledge derived from experience that has the potential for reuse in future projects.

Fig. 1. Conceptual vision of ITAKA.

ITAKA is based on the guidelines for building a PAL proposed by García et al. [35], but also includes configuration and change management principles to implement a feedback mechanism to enrich the pre-existing knowledge with experiences exchanged among software engineers even when working in different project teams or geographic locations. The following paragraphs briefly describe these new features. A formalized and detailed description of the different processes can be found in [36].

The repository of organizational knowledge (i.e. the PAL) consists of a set of assets considered useful to be instantiated and reused in different projects throughout the organization. These assets contain some explicit knowledge in the form of plain text, documents or videos, and tacit knowledge in the form of examples or discussions that have turned into lessons learned. In order to store the knowledge assets, the system has a single repository acting as a version control platform that organizes and preserves all the knowledge about the processes in the organization.
During the acquisition of the pre-existing knowledge, the organization’s experts identify knowledge obtained from the organizational software development processes, effective practices taken from the SE literature and the experience of relevant organization’s professionals. This knowledge is formalized using a pre-existing knowledge editor to define a process guide, which is registered by a librarian in the repository of organizational knowledge.

The knowledge that was previously registered is distributed by deploying a view that provides access to a version of the organizational knowledge. Since not all teams have the same needs, the librarian can deploy different types of views depending on whether the view includes functionality to collect new tacit knowledge to enrich the organizational knowledge (dynamic view) or not (static view), and also depending on whether or not the knowledge has to be available online.

The organizational knowledge can be accessed through the different views and reused by the software engineers in future projects. Either when specific information about an asset is needed or when a software engineer wants to know the next process or activity to carry out during the project execution, knowledge assets can be consulted by navigating through the view, or adapted according to the characteristics and needs of another project, thus creating an instance of that asset.

In a dynamic view of the organizational knowledge (i.e., a wiki), software engineers can make new contributions during the projects to collaboratively enrich the organizational knowledge. These contributions will be mainly in the form of examples, discussions or changes in the pre-existing knowledge. On the one hand, the discussions are composed of a heading and its corresponding text, a structure that avoids constraining its contents so that software engineers are free to debate about the knowledge of an asset, provide their experiences, describe the context, etc. On the other hand, the examples have a text describing the context to which the example belongs, a description of the problem that the example solves, the artifacts associated with the problem, a description of a solution to the problem, and the artifacts associated with that solution.

KM frameworks do not generally define the context where the experience have emerged, making it harder to identify the situations where a knowledge asset is applicable [1]. Thus, for a more effective reuse of the examples, this type of contribution first requires a de-contextualization and then a re-contextualization. In the de-contextualization, the software engineer has to isolate the example from the project in which it was produced. Then, in the re-contextualization the software engineer establishes a new context using a form provided by the wiki.

In order to upgrade the organizational knowledge with contributions from the software engineers preserved in the wiki during the execution of the projects, ITAKA implements a feedback mechanism that allows the change management of the knowledge assets stored in the repository. Changes can be managed using two alternatives. In the first one, an expert supervises every contribution to check that it is correct, consistent, relevant, and not redundant before it is stored in the repository; contributions may be discarded, edited or accepted, and only when a contribution is not discarded is it stored in
the repository of organizational knowledge. In the second one, contributions are not supervised, so they always become part of the repository of organizational knowledge. Once the repository of organizational knowledge has been updated by the librarian, a new version of the different views can be distributed to their corresponding users, which closes the feedback loop, so that the enriched organizational knowledge is available to all teams regardless of whom contributed the new knowledge.

Finally, the use of the process assets by the software engineers has to be monitored by the librarian to evaluate whether the organizational knowledge is managed properly and effectively, and to identify opportunities for improvement by introducing corrective changes in the future [35]. In addition, quality of new contributions has to be also measured as it may affect user satisfaction [37] and the success of learning will depend largely on users’ intentions to continue using the framework [38].

4. Research Design

This research aims at analyzing the influence of several factors on the learning, adoption and improvement of new SE processes and the evolution of those factors depending on the framework that supports the collaborative learning. The following sections describe the context, plan, and data gathering and analysis of the research.

4.1. Context

The research took place in a training course on agile software development with 46 junior software engineers (six women and forty men aged 22–25) divided into 12 teams. These engineers (hereafter, participants) were final-year students with a strong knowledge of SE processes and tools. All of them had similar educational background and previous experience in software development.

Students are often used instead of professionals as subjects to pilot experiments in the context of software engineering before they are carried out in industrial environments [39]. Researchers tend to use students to confirm or refute hypotheses about new technology or method adoption, and the costs and benefits of empirical studies using students are discussed in [40].

We focus our research on environments where junior software engineers have to learn and adopt new processes they might not be familiar with. Our participants were considered a close match to the population under study because their knowledge of SE practices and their performance is comparable to that of junior software engineers working in the industry. The training course is suitable for the research because it required collaboration among participants to adopt new software processes, a situation that software engineers have to face when an innovative IT company has to adapt its SE processes to the changing development restrictions, approaches and technologies.

The research was composed of two case studies (Fig. 3). In case study 1, a first basic approach to the proposed framework (named PAL-Wiki) was deployed using a wiki as a lightweight repository of knowledge about agile methods [27]. Although this framework eased the learning process by sharing the software process knowledge among engineers
and allowed them to work in a more autonomous way, it lacked support for the collaborative learning among engineers working in different project teams. In case study 2, the framework previously described in this paper (named ITAKA) was fully deployed. Instead of the single wiki used in the first case study, several wikis tailored to suit the specific needs of different project teams were used, providing all of them access to the same organizational knowledge.

The main improvement in ITAKA framework regarding PAL-Wiki was the introduction of the new feedback mechanism with change and configuration management support. It allowed gathering all the new tacit knowledge acquired by the different wikis and merging it into a single repository to enrich the organizational knowledge. The extraction of the organizational knowledge from the wiki database to an external repository and the use of the feedback mechanism allowed users of a given wiki (i.e. software engineers from a project team) to access the tacit knowledge contributed by users from other wikis (i.e. software engineers from other project teams), which was not possible with PAL-Wiki. All these architectural differences between the two frameworks are shown in Fig. 2 where components already existing in PAL-Wiki have been shaded.

In addition, the concept of “example” and the mechanism to contribute them were improved in the second case study. Instead of uploading just an archive as an example like in the first case study, users were then able to contribute enhanced examples composed of a text describing the context to which the example belonged, a description of the problem that the example solved, the artifacts associated with the problem, a description of a solution to the problem, and the artifacts associated with that solution.
4.2. Plan

The execution of the research in both case studies (Fig. 3) was divided into three stages.

Fig. 3. Research design.

The first stage (A) was the configuration of the initial knowledge available to the participants and its publication in the wikis. The wiki software (MediaWiki) and the initial knowledge was exactly the same in both case studies. The instructors carried this initial preparation out before the beginning of the training course. This stage corresponds to the knowledge acquisition and distribution activities presented in section 3.

The second stage (B) consisted of training the participants on agile software development methods. In both case studies, the training stage lasted 11 weeks and it was divided into 8 sessions: a) User stories, b) Acceptance tests, c) Coding standard, d) Simple design, e) Refactoring, f) Unit tests, g) Collective ownership and Continuous integration, h) Tracking. The participants familiarized themselves with the framework, exploring its contents to perform the practice sessions during the training.

In both case studies, instructors presented the framework to the participants at the beginning of the training stage. Periodically, instructors reminded participants how to use the framework and emphasized the importance of contributing new tacit knowledge to enhance their own learning. No additional reward was given for using the framework.

During the training stage in case study 2, the knowledge application and preservation activities presented in section 3 were carried out by the participants. At the end of the training stage in case study 2, the change management activity presented in section 3 was carried out by the instructors and a new version of the organizational knowledge was published in all the wikis as a consequence.

The last stage (C) of the research consisted of working on a project to develop a small-size software application. The project stage in both case studies started in week 12 and lasted 4 weeks. Participants had to develop similar applications in both case studies. During the project stage in case study 2, the knowledge application and preservation activities were carried out by the participants. At the end of the project stage in case study 2, the change management activity was carried out by the instructors. As a consequence, a new version of the organizational knowledge was published in all the wikis for future training courses.
4.3. **Data gathering and analysis**

This section provides specific information on the evidences and the data analysis techniques used to identify the relevant results of this research.

Statistics on user accesses and contributions to the wikis were collected during the research. These included data on access to the wikis and the number of the different types of contributions (modification of the pre-existing knowledge, examples, discussions and lessons learned) and qualitative information obtained from the instructors’ notes, observations and informal interviews to the participants. The statistics on the use of the wikis included data regarding: number of different sessions, pages visited per session, number of visits received per wiki page, number of visits received per knowledge item, number of visits to new knowledge items created, browser used in each session and the user rating of each page included in a wiki. These data were obtained using Google Analytics and MediaWiki statistics services.

Anonymous surveys were also carried out at the end of both the training and project stages (Table 1). The first survey consisted of 5 closed questions about the use of the wiki, 8 opinion questions using a 4-point Likert scale and 27 semantic differential items using a 5-point Likert scale. The second survey consisted of 5 closed questions about the use of the wiki, 12 opinion questions using a 4-point Likert scale, 30 semantic differential items using a 5-point Likert scale and 4 open questions. The closed questions asked the participant about his/her number of accesses and contributions to the wiki. The 4-point Likert scale questions assessed the ease of use of the mechanisms to acquire new knowledge and the usefulness of the new knowledge. The semantic differential items
assessed the usefulness and the quality of the artifacts. The open questions asked about the benefits and the problems encountered when using the framework and suggested improvements.

Finally, the levels of quality of the intermediate products delivered by each team during the project stage in both case studies were assessed according to objective evidences in order to discuss the improvement in the participants’ ability to learn and adopt new efficient practices for software development.

The data gathered were analyzed using a combination of regression analysis and descriptive statistics. Regression analysis allowed us to obtain insights regarding the factors that may influence the quality of the new knowledge assets (examples, discussions and lessons learned) generated using ITAKA. On the other hand, descriptive statistics helped us to understand the regression equations found in the analysis and to discuss how the factors identified influence the adoption of SE processes.

Qualitative information obtained from instructors’ notes, observations and informal interviews to the participants was also analyzed to complement and provide insight into the quantitative results.

5. Results

A regression analysis was carried out to obtain insights into the factors that may influence the quality of the knowledge assets elicited during the research and thus influence the adoption of SE processes. The explanatory variables used in this pre-analysis were: the number of new knowledge assets (i.e. number of examples, discussions and lessons learned), the simplicity of the mechanisms to contribute new knowledge assets, the number of knowledge assets accepted to become part of the repository of organizational knowledge, their usefulness for learning, their usefulness for the project, their helpfulness to solve doubts about the processes, their helpfulness to optimize the time spent on the project, and their helpfulness to deepen knowledge on SE processes.

As the feedback mechanisms to acquire new tacit knowledge were fully deployed in case study 2, only data from the second case study was considered for the regression analysis. Although sixteen software engineers participated in case study 2, three of them were the control group so they were excluded from the analysis because the wiki they accessed during the research did not implement the new feedback mechanisms (i.e. it was not possible for them to contribute examples directly through their wiki and discussions were not enabled).

The regression analysis revealed that the last three of the explanatory variables mentioned above do not influence the quality of the knowledge assets, so they were discarded. Using the remainder explanatory variables, we found the following multiple regression equations.

5.1. Analysis of the quality of examples

Considering the data in Table 2, we found a multiple regression equation (R2 = 0.9507, p-value = 0.0101) for predicting quality of examples (Qex) using the number of examples
(N), the number of accepted examples (A), their usefulness for learning (UL), their usefulness for the project (UP) and the simplicity of the mechanism (S).

\[
Qex = -1.85700 + 0.95444 \cdot N + 0.67793 \cdot UL + 0.89276 \cdot UP + 0.01562 \cdot S - 0.87471 \cdot A
\]

5.2. Analysis of the quality of discussions

Considering the data in Table 3, we found a multiple regression equation (R² = 0.8149, p-value = 0.0448) for predicting quality of discussions (Qd) using the number of discussions (N), their usefulness for learning (UL), their usefulness for the project (UP) and the simplicity of the mechanism (S).

\[
Qd = -0.47970 + 0.19890 \cdot N + 1.88260 \cdot UL + 0.27361 \cdot S - 0.86165 \cdot UP
\]

Table 2. Subjective evaluation of the examples at the end of the project stage in case study 2.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Qex</th>
<th>N</th>
<th>A</th>
<th>UL</th>
<th>UP</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5.00</td>
<td>9</td>
<td>9</td>
<td>4.00</td>
<td>3.75</td>
<td>2.00</td>
</tr>
<tr>
<td>P2</td>
<td>4.00</td>
<td>10</td>
<td>10</td>
<td>3.20</td>
<td>2.80</td>
<td>4.00</td>
</tr>
<tr>
<td>P3</td>
<td>5.00</td>
<td>10</td>
<td>10</td>
<td>4.00</td>
<td>2.50</td>
<td>n/a</td>
</tr>
<tr>
<td>P4</td>
<td>4.00</td>
<td>9</td>
<td>9</td>
<td>3.20</td>
<td>3.55</td>
<td>2.00</td>
</tr>
<tr>
<td>P5</td>
<td>4.00</td>
<td>10</td>
<td>10</td>
<td>4.00</td>
<td>2.55</td>
<td>2.00</td>
</tr>
<tr>
<td>P6</td>
<td>5.00</td>
<td>9</td>
<td>9</td>
<td>4.00</td>
<td>4.00</td>
<td>2.00</td>
</tr>
<tr>
<td>P7</td>
<td>4.00</td>
<td>10</td>
<td>10</td>
<td>3.20</td>
<td>3.55</td>
<td>4.00</td>
</tr>
<tr>
<td>P8</td>
<td>4.00</td>
<td>9</td>
<td>9</td>
<td>2.40</td>
<td>3.50</td>
<td>2.00</td>
</tr>
<tr>
<td>P9</td>
<td>n/a</td>
<td>10</td>
<td>10</td>
<td>n/a</td>
<td>n/a</td>
<td>3.00</td>
</tr>
<tr>
<td>P10</td>
<td>4.00</td>
<td>10</td>
<td>9</td>
<td>2.40</td>
<td>2.60</td>
<td>1.00</td>
</tr>
<tr>
<td>P11</td>
<td>4.00</td>
<td>10</td>
<td>10</td>
<td>4.00</td>
<td>n/a</td>
<td>2.00</td>
</tr>
<tr>
<td>P12</td>
<td>4.00</td>
<td>10</td>
<td>10</td>
<td>3.20</td>
<td>3.25</td>
<td>1.00</td>
</tr>
<tr>
<td>P13</td>
<td>1.00</td>
<td>10</td>
<td>9</td>
<td>0.80</td>
<td>0.95</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 3. Subjective evaluation of the discussions at the end of the project stage in case study 2.

<table>
<thead>
<tr>
<th>Participant</th>
<th>Qd</th>
<th>N</th>
<th>UL</th>
<th>UP</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5.00</td>
<td>0</td>
<td>4.00</td>
<td>3.80</td>
<td>4.00</td>
</tr>
<tr>
<td>P2</td>
<td>3.00</td>
<td>0</td>
<td>2.40</td>
<td>2.80</td>
<td>3.00</td>
</tr>
<tr>
<td>P3</td>
<td>3.00</td>
<td>0</td>
<td>2.40</td>
<td>2.40</td>
<td>3.00</td>
</tr>
<tr>
<td>P4</td>
<td>4.00</td>
<td>0</td>
<td>3.20</td>
<td>n/a</td>
<td>3.00</td>
</tr>
<tr>
<td>P5</td>
<td>3.00</td>
<td>0</td>
<td>3.20</td>
<td>2.80</td>
<td>3.00</td>
</tr>
<tr>
<td>P6</td>
<td>5.00</td>
<td>8</td>
<td>2.40</td>
<td>2.35</td>
<td>3.00</td>
</tr>
<tr>
<td>P7</td>
<td>3.00</td>
<td>1</td>
<td>2.40</td>
<td>2.60</td>
<td>3.00</td>
</tr>
<tr>
<td>P8</td>
<td>1.00</td>
<td>0</td>
<td>1.60</td>
<td>2.60</td>
<td>2.00</td>
</tr>
<tr>
<td>P9</td>
<td>n/a</td>
<td>1</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>P10</td>
<td>2.00</td>
<td>0</td>
<td>1.60</td>
<td>1.90</td>
<td>n/a</td>
</tr>
<tr>
<td>P11</td>
<td>4.00</td>
<td>10</td>
<td>1.60</td>
<td>2.35</td>
<td>4.00</td>
</tr>
<tr>
<td>P12</td>
<td>2.00</td>
<td>10</td>
<td>1.60</td>
<td>2.85</td>
<td>3.00</td>
</tr>
<tr>
<td>P13</td>
<td>2.00</td>
<td>0</td>
<td>1.60</td>
<td>1.45</td>
<td>4.00</td>
</tr>
</tbody>
</table>

\[
Qd = -0.47970 + 0.19890 \cdot N + 1.88260 \cdot UL + 0.27361 \cdot S - 0.86165 \cdot UP
\]
5.3. Analysis of the quality of lessons learned

Considering the data in Table 4, we found a multiple regression equation ($R^2 = 0.9547$, p-value = 0.0003588) for predicting quality of lessons learned (Qll) using the number of discussions in the training stage that became lessons learned in the project stage (N), their usefulness for learning (UL), their usefulness for the project (UP), and the number of new discussions in the project stage (ND).

$$Qll = 1.92123 + 0.21960 \cdot UL + 0.28930 \cdot UP - 0.02057 \cdot N - 0.15265 \cdot ND$$  \hspace{1cm} (3)

Observing the regression equations found in this pre-analysis and considering the qualitative information obtained from participants in the case studies, we identified five influential factors on the quality of the knowledge assets elicited during the research and thus on the adoption of software engineering processes:

1. The commitment of software engineers to collaborate in the learning process, as their participation in the collaborative learning (i.e. number of contributions) influences their quality.
2. The perceived level of usefulness of the tacit knowledge elicited during the learning process, as the usefulness of contributions for learning and for the project influences their quality.
3. The diversity of the topics covered by the shared knowledge, as the number of new contributions and the number of accepted contributions influence their quality depending on whether they represent a wide range of situations or not.
4. The simplicity of the mechanisms to contribute new tacit knowledge, as it has direct influence on the quality of contributions.
5. The amount of learning achieved by software engineers, which was not directly extracted from any of the regression equations found, but the enhancement of the learning and adoption of new processes is the purpose of the proposed framework.
6. Factors Influencing the Adoption of Software Engineering Processes

This section analyses the influence of five factors (commitment, usefulness, diversity, simplicity and learning) on the adoption of SE processes. To do so, results from the surveys carried out in the two case studies presented previously (PAL-Wiki and ITAKA) are compared. Statistical analysis of those results will provide further insights into the influence of the factors on the quality of the knowledge assets elicited during the research and thus their influence on the adoption of SE processes.

Table 5. Summary of findings during the research.

<table>
<thead>
<tr>
<th></th>
<th>Case Study 1 (PAL-Wiki)</th>
<th>Case Study 2 (ITAKA)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commitment</strong></td>
<td>- Participants were not committed towards a collaborative learning because they were worried about their own learning</td>
<td>- The new framework stimulated the collaboration in the learning process</td>
</tr>
<tr>
<td></td>
<td>- Participants consulted knowledge but did not contribute new experiences</td>
<td>- The lack of involvement of some participants affected negatively the learning process of others</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Participants actively involved in contributing new tacit knowledge adopted better the new SE processes</td>
</tr>
<tr>
<td><strong>Usefulness</strong></td>
<td>- Only the pre-existing knowledge was considered useful to learn better and produce better quality products</td>
<td>- Pre-existing knowledge was considered useful to learn better and produce better quality products</td>
</tr>
<tr>
<td></td>
<td>- The usefulness of new contributions was not clear</td>
<td>- New tacit knowledge enhanced the pre-existing knowledge and was useful for learning and problem solving</td>
</tr>
<tr>
<td><strong>Diversity</strong></td>
<td>- A large number of examples were related to a very specific topic, which was not considered useful by participants</td>
<td>- A large number of examples were related to a very specific topic, which was not considered useful by participants</td>
</tr>
<tr>
<td></td>
<td>- Participants demanded more previous experiences in the form of examples and discussions</td>
<td>- Participants demanded more previous experiences in the form of examples, discussions and lessons learned</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- An expert supervising the new contributions helps to maintain diversity</td>
</tr>
<tr>
<td><strong>Simplicity</strong></td>
<td>- The mechanism to participate in discussions was very easy to use, although it did not stimulate participation</td>
<td>- The mechanism to participate in discussions was very easy to use</td>
</tr>
<tr>
<td></td>
<td>- The mechanism to upload artifacts as examples was very easy to use</td>
<td>- The mechanism to contribute new examples was more complex due to the improved concept of “example”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Participants required some time to use these mechanisms effectively</td>
</tr>
<tr>
<td><strong>Learning</strong></td>
<td>- The framework eased the process of learning agile practices</td>
<td>- The new framework enhanced the ability of participants to collaboratively learning</td>
</tr>
<tr>
<td></td>
<td>- The framework allowed participants to work in a more autonomous way</td>
<td>- The amount of new knowledge assimilated by participants must fulfill their expectations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The more knowledge a participant shared, the more he/she expected from others</td>
</tr>
</tbody>
</table>
Table 5 gathers the results obtained in both case studies regarding the five factors previously identified, which will be further discussed in the following sub-sections. The order in which the five factors are presented is not especially relevant.

6.1. Commitment

The commitment of software engineers to collaborate in the learning process is related to the amount of new tacit knowledge provided by them to enrich the organizational knowledge and is a key issue in the adoption of new SE processes in any organization.

In case study 1, participants were not totally committed towards a collaborative learning of the new agile methods. The statistical analysis of data gathered through the surveys in case study 1 revealed that almost 70% of the participants considered that neither the PAL-Wiki framework promoted the creation of a community of users of the software development process nor it stimulated them to share knowledge with other software engineers. They preferred sharing new knowledge with their project colleagues using face-to-face conversations rather than using the PAL-Wiki framework to share new knowledge with other project teams. As a consequence, they used the PAL-Wiki framework to consult knowledge rather than to contribute new experiences because they were more worried about their own learning than about helping others to learn. Most of them surprisingly stated in the surveys that it would be interesting if more previous experiences were available for them.

The new feedback mechanisms of the ITAKA framework deployed in case study 2 enhanced the collaborative learning among engineers working in different project teams. Data gathered through the surveys and other qualitative information obtained from the instructors’ notes, observations and informal interviews to the participants confirmed the need for commitment of software engineers. Some participants mentioned that the lack of participation of some colleagues affected negatively the overall quality of the knowledge assets. In addition, the instructors observed that the participants who contributed more actively were more likely to produce better quality products; we do not have, however, enough data to statistically prove this statement, so future studies should consider a deeper analysis of this hint.

Finally, the equations included in section 5 of this paper strengthen the importance of the commitment of software engineers to collaborate in the learning process with contributions that enrich the organizational knowledge. The more knowledge assets software engineers share, the better the quality of those assets is.

Eq. (1) and (2) showed that the number of examples and discussions is an enabler of their respective quality \(Q_{ex} \propto 0.95444\cdot N\); \(Q_d \propto 0.19890\cdot N\). When the number of contributions in the form of examples is high, there are more instances of different situations available to the software engineers, which eventually improve the overall perception of their quality. Discussions are however a good mechanism to encourage collaboration among the members of a work team through the generation of a common knowledge base. The more engineers contribute to the discussions, the more the team
works together to overcome the challenges in the project, resulting in a higher consideration of the quality of the discussions.

Eq. (3) revealed the number of discussions in the training stage that became lessons learned in the project stage and the number of new discussions in the project stage as inhibitors of the quality of the lessons learned (Qll ∝ - 0.02057·N - 0.15265·ND). Although their coefficients are low, they may influence the perception of quality to some extent. Their negative signs show that the quality of the lessons learned can be high only if not all the discussions are accepted to become lessons learned. This finding suggests that the role of the supervised change management is essential because not all the discussions should be accepted in the feedback process; this result is consistent with the conclusions in [1] that recommended the verification of the new contributions before they are stored in the repository for maintaining system reliability and effectiveness. When the supervision is rigorous, only the best discussions become part of the organizational repository, so the quality of the consequent lessons learned is better.

6.2. Usefulness

Another factor that influences the adoption of SE processes is the perceived level of usefulness of the tacit knowledge elicited during the learning process.

In case study 1, the pre-existing knowledge stored in the PAL-Wiki framework was considered useful by the participants as it helped them to learn better and produce better quality products. Data from the surveys also revealed that the most useful pre-existing knowledge assets were the examples of previous experiences existing in the wiki (rated 4.47 on a 1-to-5 scale). But regarding new contributions participants made to the wiki, only 7.7% of them considered new contributions to be useful for learning agile methods, and 65.4% of them did not even have a clear opinion about the usefulness of those contributions. The problem was that the new artifacts obtained in case study 1 that could have been used as new examples were stored in an external repository and not in the wiki itself. This fact made participants could not have an immediate access to those new examples making them difficult to reuse, so participants call the usefulness of new contributions into question as a consequence.

This issue did not appear in case study 2 because examples were stored directly in the wiki itself. Results obtained from the surveys in case study 2 showed that participants perceived the elicited tacit knowledge —mainly in the form of examples— to be useful for the learning and adoption of the new SE processes. Aspects such as its usefulness for learning and problem solving were better rated than other aspects such as its usefulness for developing products faster or better understanding the business context. These findings suggest that organizations should carefully analyze the situation if they plan to use such frameworks for activities more complex than learning and problem solving.

The equations presented in section 5 indicated that there is also a positive relation between the usefulness of knowledge assets and their quality. Regarding the examples, Eq. (1) highlighted the influence of their usefulness for learning the processes during the training stage and also for adopting and putting them into practice during the project.
Factors Influencing the Adoption of Agile Processes When Using Wikis

Stage (Qex \( \propto 0.67793 \cdot UL + 0.89276 \cdot UP \)). Usefulness of examples for learning is an enabler of their quality because they helped participants to learn how to apply their knowledge to deal with different problems in future projects. Even when participants were not able to find an adequate example, consulting other examples helped them to learn how to carry out a process. In addition, similar problematic situations occur in different projects and the examples provided the knowledge required to tackle those problems during the project stage. Thus, it is essential to describe correctly the problem that the example solves, granting its independence from the characteristics of the project. This is the reason why the process of de-contextualization and re-contextualization is fundamental to facilitate an efficient search for examples by other participants.

Eq. (2) shows a positive influence of the usefulness of discussions during training stage but a negative influence during the project one (Qd \( \propto 1.88260 \cdot UL - 0.86165 \cdot UP \)). Usefulness of discussions during the learning stage had a strong and positive influence in the perception of their quality, while their usefulness during the project stage had a negative effect. Discussions provided a mechanism to solve questions related to a process during the training stage, so they were considered useful because they helped to learn and understand that process and the quality perceived by the users improved as a consequence. But during the project stage participants considered discussions were an adequate mechanism to solve questions about the requirements of the projects. As a result, discussions did not help to better learn and adopt the processes and therefore they were not considered as useful as the examples.

Regarding lessons learned, their usefulness for learning and for the project stimulated the perception of their quality according to Eq. (3) (Qll \( \propto 0.21960 \cdot UL + 0.28930 \cdot UP \)). Even more, this positive influence was stronger than the negative influence of the number of contributions (Qll \( \propto -0.02057 \cdot N - 0.15265 \cdot ND \)), so in the case of the lessons learned the usefulness is more important than the quantity. Lessons learned help consolidate the new knowledge that comes from discussions. This consolidation was found to be a key process in increasing the quality of the lessons learned. It was found however that the participants would have liked to know more about the criteria applied to promote a discussion to a lesson learned. They argued that these criteria should have been public and more transparent. The participants who contributed to the discussions were eager to know why their discussions were considered or not as lessons learned.

6.3. Diversity

A factor related to the usefulness of knowledge assets is the diversity of the topics covered by the shared knowledge. Commitment of software engineers towards participation in the learning process can increase the number of contributions to enrich the organizational knowledge, but increasing the amount of knowledge assets related to a very specific topic may reduce the level of usefulness and quality perceived by software engineers.

Participants in both case studies demanded more previous experiences in the form of examples and lessons learned because they were considered useful for learning and
understanding the processes, and also helped them to deal with the problems that arise over the course of a project. During the research, there were a large number of examples related to a very specific topic, whereas few or none were related to others. Participants declared that similar examples were not considered useful because they were repeated.

The regression equations (1) and (3) in section 5 support this statement that not all contributions might help to increase the perceived level of quality of the knowledge assets. On the one hand, Eq. (1) indicated that the number of accepted examples influenced their quality in a negative manner ($Q_{ex} \propto -0.87471 \cdot A$). On the other hand, Eq. (3) showed that the number of discussions in the training stage that became lessons learned in the project stage was found to be an inhibitor of the quality of the lessons learned ($Q_{ll} \propto -0.02057 \cdot N$).

It becomes important, therefore, to identify ways to elicit examples that represent a wide range of situations in order to not reduce the overall perception of quality of the knowledge assets. Likewise, it is important to encourage discussions related to different topics. When too many discussions are centered on specific aspects of a particular project, the quality perceived by other users is reduced because not all the participants may find them useful for other projects.

Some situations that may reduce the perceived level of quality, such as repeated examples, discussions associated with an inappropriate asset, low-quality discussions becoming lessons learned, and so on, can be avoided if an expert supervises the new contributions prior to them becoming part of the organizational repository. The expert may edit —if necessary— a contribution to adapt it to the required level of quality or even discard it when it cannot be adapted. The supervised change management included in the ITAKA framework (case study 2) has proven to be a useful mechanism to filter contributions according to their quality.

6.4. Simplicity

Simplicity is related to the complexity of the mechanisms to provide new knowledge, and it needs to be a subjective measure from the point of view of the users of those mechanisms.

In case study 1, the mechanisms to elicit new knowledge from software engineers were very easy to use. On the one hand, discussions mechanism was a functionality directly provided by the wiki software (MediaWiki) consisting of a wiki page (talk page) associated to the corresponding knowledge asset in which users could add comments to share knowledge with other users. This mechanism was strange for some participants because it did not follow the conventional format of forums, but it gave the opportunity to create the sense of collective ownership of the organizational knowledge within the community of software engineers as anyone could enrich existing discussions or start new ones. Although participants did not make the most of this mechanism —because they preferred face-to-face conversations as it was mentioned before—, up to 86.7% of them stated the mechanism was easy to use. On the other hand, artifacts produced during the research were uploaded by software engineers to an external repository to be
available as examples for future projects. Surveys shown 80.8% of participants considered the external repository adequate for storing artifacts.

In case study 2, ITAKA framework used the same discussions mechanism than in case study 1, so it was logical to obtain similar results in the surveys (83.3%) regarding its simplicity. Despite its simplicity, some participants initially refused to use discussions. It should be taken into account that the environment moved from informal face-to-face conversations and e-mail (case study 1) to a common knowledge base created using discussions in wikis (case study 2). Familiarizing with the new environment required some time for the participants in case study 2, who gradually used discussions more. On the other hand, the mechanism to contribute new tacit knowledge in the form of examples was different because they were stored directly in the wiki instead of in an external repository. Furthermore, an improved concept of “example” was used, so examples did not consist of just an archive like in the first case study, but a text describing the context to which the example belonged, a description of the problem that the example solved, the artifacts associated with the problem, a description of a solution to the problem, and the artifacts associated with that solution. This difference obviously made contributing examples to be more laborious because the process of de-contextualization and re-contextualization required an extra effort. According to [10] wikis can be unintuitive, so providing users with some guidelines may help to ease the use of this mechanism to contribute new examples.

The regression equations in section 5 revealed that simplicity of the mechanisms can influence the quality of the knowledge assets in a positive manner. Its influence was, however, stronger in the case of the discussions than in the case of examples, as their coefficients in equations (1) and (2) showed ($Q_{ex} \propto 0.01562 \cdot S ; Q_{d} \propto 0.27361 \cdot S$). The mechanism to contribute new discussions was considered easier to use than one to contribute new examples, so it stimulated the quality of the discussions in a higher degree.

6.5. Learning

Learning refers to the amount of new knowledge assimilated by the participants and can be measured through the evaluation of the levels of quality of the intermediate products delivered by each team during the project stage in both case studies.

Table 6 compares the levels of quality of the products developed during the project stage with ITAKA (case study 2), with PAL-Wiki (case study 1) and without any framework to support the learning. These products were evaluated according to objective evidences (already published in [34] and [35]) and their analysis revealed that PAL-Wiki eased the learning process and allowed software engineers to work in a more autonomous way [35]. On the other hand, ITAKA proved to be effective in enhancing the ability of software engineers to learn SE best practices to some extent [34] because the product quality improved in those cases which required a great deal of tacit knowledge.
Data gathered through surveys in the scope of this research allow us to further discuss the influence of the amount of learning achieved by software engineers on the adoption of SE processes. All participants in case study 1 agreed that PAL-Wiki framework eased the process of learning agile process and 80.8% stated that it provided an adequate environment to learn new SE processes. Regarding the new mechanisms to elicit new tacit knowledge in case study 2, as it was previously mentioned, discussions were rated lower than examples, and less than half of the participants considered discussions helped them to solve doubts about the processes or to deepen knowledge on SE processes. This fact was due to the use participants sometimes made of discussions to ask for solutions to specific problems about the projects instead of using them to discuss about the processes themselves.

The surveys in case study 2 also revealed that there can be a negative influence in the perception of quality of the elicited tacit knowledge related to expectations when referring to individuals. Participants who contributed very actively to the discussions since the beginning of the research evaluated less positively the contributions to the discussions from other participants. It could be due to the fact that the more they participate, the more knowledge they assimilate and the more they expect from others. Consequently, if the knowledge contributed from others is poorer than the knowledge they contributed, the overall quality of the knowledge perceived is lower.

7. Threats to Validity

This section discusses the threats to validity and limitations of the study presented in this paper. As described by Wohlin et al. in [41], validity can be discussed in terms of construct, internal, external, and conclusion validity.

7.1. Construct validity

Construct validity refers to the accuracy of the research strategy and variables considered for discussing appropriately the research questions previously stated.
Some of the data considered in the analysis are based on the statistics provided by Google Analytics, which monitors some of the variables considered in the definition of the research. Nevertheless, Google Analytics have several limitations that prevent the acquisition of additional information such as the identification of the user who is accessing the wiki services in each session. This information could have been very useful to statistically analyze in depth some evidences related to the behavior of the software engineers participating in the research. This limitation was partially solved however by the instructors’ notes, observations and informal interviews to the participants during the research execution.

The surveys were helpful to analyze the evolution of the attitude and aptitude of software engineers when using the framework presented in this article to learn and adopt new processes. Nevertheless, additional quantitative information could have been useful to analyze statistically the influence of human related aspects such as team work and culture of knowledge sharing. Those aspects would provide a more holistic view on the factors influencing the adoption of new SE processes.

7.2. **Internal validity**

Internal validity refers to the degree to which statements about cause and effect are valid.

In this research work, the researchers’ opinion was not imposed on the participants. However, it is impossible not to influence the participants to some degree because instructors’ opinions might be presupposed just by guiding the research. The anonymity of the surveys likely reduced this bias as participants were free to express their own opinions and to provide honest answers.

7.3. **External validity**

External threats to validity limit the degree to which generalizations can be drawn from our results.

Several contextual limitations regarding the number of participants, number and type of projects developed, and the participants’ previous experience can be envisaged. This research tries to contribute to the discussion on the factors that influence the learning and adoption of new processes in common software development contexts, and not to create a general theory from the findings. Typically, software development units work in reduced types of projects centered on specific businesses and technologies. When large organizations want to diversify their business, they usually create independent units. Even more, participants from our research had similar profiles to personnel working for innovative IT companies with little background in software development processes and information modeling. So these contextual restrictions can be considered as representative of a relevant set of SE organizations.

An important threat to external validity that we must highlight concerns the characteristics of the software engineers participating in the research. The participants were undergraduates, so they were not necessarily representative of all junior software engineers. However, most of them had real experience in activities related to IT — which
is typical in Spain where undergraduates usually work while finishing their BSc—, and their responses were generally aligned with those from graduate students.

Finally, it is important to remark that one of the aspects that contribute to the effective use of the ITAKA framework is the existence of a culture of collaboration and knowledge sharing among the software engineers working within the organization to elicit best practices for developing software products. The context of this empirical study was characterized by the willingness of the participants towards sharing knowledge. Nevertheless, the results could be different if there is a strong competitiveness among software engineers and a lack of knowledge sharing culture.

The empirical study described was performed at Carlos III University and we have observed a high degree of collaboration among the students; this collaboration fostered knowledge sharing, which improved the quality of the new knowledge assets created. We cannot guarantee the same results if the working environment is not highly collaborative.

7.4. Conclusion validity

Conclusion validity is the degree to which conclusions made based on the findings were reasonable. It is concerned with the relationship between the treatment and the outcome of the research, being a treatment one particular value of a factor. In our study, the factor is the framework to enhance collaborative learning of SE processes, so the treatments are the old framework (PAL-Wiki) and the new one (ITAKA, described in section 3).

The multiple regression equations found provide statistically significant results with regard to the factors influencing the quality of the new knowledge elicited in the form of examples, discussions and lessons learned when using the ITAKA framework. The coefficients of determination $R^2$ for these equations indicate that the regression lines approximate the real data points.

Another threat to the conclusion validity is the reliability of measures. As most of the data in the research came from the subjective opinion from participants gathered through surveys, repeating the research under the same conditions might not provide the same results. However, this research do not try to infer a theory from findings but to contribute to understand how KM can help to train on use of new processes more effectively by transferring knowledge from experienced software engineers to less experienced ones using social software.

8. Summary and Conclusions

This paper proposes a framework (named ITAKA) based on social software to support the collaborative learning, adoption and improvement of SE processes through the exchange of experiences among individuals. The research analyses the influence of five factors (commitment, usefulness, diversity, simplicity and learning) on the adoption of new SE processes and the quality of the experiences shared using ITAKA in comparison with similar frameworks.

Results obtained from the analysis of data from the two case studies within this research revealed that the factors mentioned above can stimulate or inhibit the adoption
Factors Influencing the Adoption of Agile Processes When Using Wikis

of new SE processes in different manners. First, software engineers must commit to collaborate in the learning process in order to stimulate the effective adoption of new SE processes in an organization. The lack of participation of some software engineers may affect negatively the learning process of other colleagues. Furthermore, software engineers actively involved in contributing new tacit knowledge during the learning process are more likely to adopt better the new SE processes. On the other hand, consulting previous experiences helps software engineers in the learning process, but there cannot be such experiences if other software engineers were not previously committed to contribute them in the past.

Another factor that stimulates the effective adoption of new SE processes is the perceived level of usefulness of the new experiences elicited during the learning process. That new tacit knowledge must be considered useful by software engineers to help them learn better and produce better quality products. In the scope of this research, examples of previous experiences were considered more useful than discussions and lessons learned. These types of tacit knowledge shared during the learning process are useful for software engineers to learn how to apply their knowledge to deal with different problems in future projects rather than to develop products faster or better understand the business context.

Those new experiences elicited during the learning process must also represent a wide range of situations to be considered useful and helpful. When there is no diversity on the topics covered by the tacit knowledge shared during the learning process, the level of usefulness and quality perceived by software engineers is reduced, which inhibits the adoption of the new SE processes as a consequence. An expert supervising the new contributions from software engineers prior to them becoming part of the organizational repository helps to maintain diversity and avoid situations that may influence negatively the learning process, such as repeated examples, discussions associated with an inappropriate asset, low-quality discussions becoming lessons learned, and so on.

The fourth factor analyzed in this research was the simplicity of the mechanisms to contribute new tacit knowledge. Complex mechanisms can inhibit the adoption of new SE processes because software engineers may not use them effectively, so the quality of shared knowledge can be therefore reduced. Some mechanisms may appear to be complex to software engineers at first, so some time is needed to use them effectively. Providing users with some guidelines also helps to ease the use of these mechanisms.

Finally, the amount of new knowledge assimilated by software engineers must fulfill their expectations in order to stimulate the effective adoption of new SE processes. The more knowledge a software engineer shares during the learning process, the more knowledge he/she expects from others. Thus, if the knowledge contributed from others is poorer, the overall perceived quality of shared knowledge is lower and the amount of learning achieved is reduced.

All these factors highlight important areas for future researchers and practitioners and should be taken into account when developing collaborative learning environments in SE.

To conclude the article, we must highlight the fact that this study tries to contribute to the discussion on the factors that influence the learning and adoption of new software
development processes, and not to create a general theory from the results. Although it can be argued that our participants in the training course are similar but not representative of the software engineers community working in the industry, the evidence gained here provides valuable insight for the industry into the factors influencing the adoption of innovative SE processes. Even though it is uncertain to which other types of problems it is possible to generalize our findings, we think that the results are at least representative for the learning and adoption of agile processes when using wikis.

This is an exploratory study and more case studies with a larger number of subjects will be necessary to confirm our findings in other settings. Our current and future research aims at carrying out similar case studies in several organizations focused on mobile application development with a larger sample size.

References


