Information and Software Technology 112 (2019) 68–82

Contents lists available at ScienceDirect

Information and Software Technology

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

Towards a reduction in architectural knowledge vaporization during agile global software development

Gilberto Borrego[a,*], Alberto L. Morán[a], Ramón R. Palacio[b], Aurora Vizcaíno[c], Félix O. García[c]

[a] Universidad Autónoma de Baja California, Mexico
[b] Instituto Tecnológico de Sonora, Mexico
[c] Universidad Castilla-La Mancha, Spain

ARTICLE INFO

Keywords:
Agile global software development
Architectural knowledge
Knowledge vaporization
Documentation debt

ABSTRACT

Context: The adoption of agile methods is a trend in global software development (GSD), but may result in many challenges. One important challenge is architectural knowledge (AK) management, since agile developers prefer sharing knowledge through face-to-face interactions, while in GSD the preferred manner is documents. Agile knowledge-sharing practices tend to predominate in GSD companies that practice agile development (AGSD), leading to a lack of documents, such as architectural designs, data models, deployment specifications, etc., resulting in the loss of AK over time, i.e., it vaporizes.

Objective: In a previous study, we found that there is important AK in the log files of unstructured textual electronic media (UTEM), such as instant messengers, emails, forums, etc., which are the preferred means employed in AGSD to contact remote teammates. The objective of this paper is to present and evaluate a proposal with which to recover AK from UTEM logs. We developed and evaluated a prototype that implements our proposal in order to determine its feasibility.

Method: The evaluation was performed by conducting a study with agile/global developers and students, who used the prototype and different UTEM to execute tasks that emulate common situations concerning AGSD teams' lack of documentation during development phases.

Results: Our prototype was considered a useful, usable and unobtrusive tool when retrieving AK from UTEM logs. The participants also preferred our prototype when searching for AK and found AK faster with the prototype than with UTEM when the origin of the AK required was unknown.

Conclusion: The participants' performance and perceptions when using our prototype provided evidence that our proposal could reduce AK vaporization in AGSD environments. These results encourage us to evaluate our proposal in a long-term test as future work.

1. Introduction

Agile and global software development (AGSD) is currently an important trend [1]. In fact, VersionOne of the 11th annual state of agile report[1] states that 86% of the respondents had distributed teams practicing agile software development (ASD). AGSD leads to many challenges, given the inherent nature of both paradigms: ASD and global software development (GSD). On the one hand, GSD communication is commonly based on documents, i.e.,explicit knowledge, that decrease the effect of the four distances of this paradigm (physical, temporal, linguistic and cultural) [2]. On the other, the agile manifesto [3] states that in ASD, face-to-face interactions are preferable to following a strict communication processes, and working software is preferable to comprehensive documentation, leaving the interpretation of the term "comprehensive" to each agile team [4]. In fact, ASD suggests that most documentation can be replaced by enhancing informal communication, i.e., a stronger emphasis on tacit knowledge rather than explicit knowledge [5]. However, prioritizing communication in ASD does not mean disregarding formal documentation [6]. This shows an internal antagonism within AGSD, since tacit knowledge is preferred in ASD (face-to-face interaction) and explicit knowledge (based on documents) is preferred in GSD.

In AGSD teams, tacit knowledge tends to predominate over explicit knowledge [6–8], leading to a lack of documents concerning architectural design, user manuals, data models, updated requirements specification, etc., known as documentation debt [9]. AGSD teams are affected by documentation debt, particularly when there is insufficient explicit

* Correspondence author.
E-mail address: gilberto.borrego@uabc.edu.mx (G. Borrego).

[1] https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2.

https://doi.org/10.1016/j.infsof.2019.04.008
Received 23 June 2018; Received in revised form 22 March 2019; Accepted 17 April 2019
Available online 23 April 2019

architectural knowledge (AK). AK is composed of architectural design (including fundamental system concepts in its environment, embodied in its elements, relationships, and in the principles of its design and evolution [10]) and of the design decisions and rationale used to attain architectural solutions [11].

One of the main problems in GSD is generally the lack of explicit knowledge (including AK) when stakeholders attempt to resolve previously presented problems, especially when this occurs in small and medium companies [12,13]. The most significant causes of a lack of explicit AK in AGSD teams are: (1) the most popular agile methods,[2] Scrum and XP [14,15], specify AK in a very lax manner, leading to documents with informal notations [4]; (2) the inherent time pressures of ASD cause the omission of appropriate documentation [16], and (3) agile developers consider that documentation is a secondary and non-creative activity [17].

In co-located ASD, the lack of AK documentation is mitigated by developers' daily face-to-face interactions. In AGSD teams, however, the lack of AK documentation is often mitigated by communicating with remote teammates using unstructured textual electronic media (UTEM), such as emails, forums, comments boards, instant messenger, etc., mainly because UTEM reduce the language gap [18]. If remote teammates are unavailable or are unable to answer their questions, agile/global developers usually attempt to obtain answers by analyzing source code [19], which is time consuming. Furthermore, the knowledge obtained is generally unstructured, incomplete and inconsistent [20], which does not guarantee that the software will evolve as planned at design time.

Furthermore, literature reports that UTEM logs contain important AK for agile/global developers [19,21], but that is unstructured, inaccessible, dispersed and prone to be lost over time, i.e., prone to be vaporized [22]. Moreover, in AGSD teams, requirements and user stories are usually the only documented knowledge referring to software development tasks [19]; there are also informal diagrams, but they are created only as an aid to problem understanding and are, therefore, considered as disposable documents. Furthermore, agile/global developers usually attempt to obtain AK from UTEM logs to mitigate this lack of documentation [19]. However, the problem is that UTEM are not designed to search AK, and developers usually use more than one UTEM to share knowledge, signifying there is no single point at which to find AK. It is, therefore, important that agile/global developers have efficient means to access the AK in UTEM logs to reduce AK vaporization.

In this paper, we present the AK Condensation concept, conceived as a means to reduce AK vaporization in AGSD by taking advantage of the knowledge stored in UTEM logs, and by giving agile/global developers the means to search for the AK contained in the aforementioned logs at a single point. This concept was implemented in a tool evaluated by agile/global developers and students to determine its feasibility. The remainder of this paper is organized as follows: Section 2 presents the related works, while the concept of AK Condensation and its implementation are presented in Section 3. Section 4 describes the evaluation method, while Section 5 presents the evaluation results and Section 6 shows the threats to validity. Finally, a discussion of the results and our conclusion are presented in Sections 7 and 8, respectively.

2. Related work

2.1. Architectural knowledge management in agile and global software development

Knowledge management is currently an important part of any software development process. Dalkir [23] proposed that KM consists of creating/capturing, sharing/disseminating and acquiring/applying knowledge assets, where: creating/capturing refers to developing new knowledge from experience and/or explicit knowledge, and then coding the knowledge in an agreed format; sharing/disseminating refers to storing knowledge in a common repository, sending it to the appropriate people or sharing it during a training session, and acquiring/applying refers to the learning process and using new knowledge in practice, with the possibility of creating knowledge to start the cycle again. This definition could, therefore, be adapted to define AKM as the discipline of creating/capturing, sharing/disseminating and acquiring/applying a software process's AK assets. This adaptation is very close to Farenhorst and de Boer [24] AKM's definition, which states that the aim of AKM is to codify software architects' tacit knowledge explicitly in either structured or semi-structured knowledge bases.

KM is a challenge in AGSD [25–27], signifying that AKM is also a challenge. A critical part of AKM is the process of knowledge capturing, because the AGSD environment [16] and the agile developers' attitudes [17] cause documentation debt [9]. Since an AGSD environment leads to a lack of captured AK, then the AKM phases of sharing/disseminating and acquiring/applying are also affected, because AK is shared and acquired on the basis of inappropriate documentation or even tacit knowledge.

Several works address AKM in software engineering [28–32], in GSD [33–37], and even in ASD [6,38,39]; however, these works do not cover AGSD environments. We, therefore, conducted a systematic mapping review (reported elsewhere [40]), in which we identified nine approaches used to manage AK that were grouped into three areas: (1) artifact-based, (2) communication-based, and (3) methodology-based. The artifact-based documentation area refers to the use of software development support (repositories, wikis and groupware) to share AK, auto-generated documentation based on communication analysis (relating to emails and code repositories), and lightweight approaches to register architecture designs and decisions. The communication-based area refers to the use of videoconference and UTEM to discuss and share knowledge, and the use of smartboards or electronic displays to show information about project architecture. The methodology-based area refers to agile method modification by introducing an architecting phase or an architect role to manage projects' AK.

We additionally observed that the papers reviewed evenly support the three phases of the integrated KM cycle [23] (Capture/Creation – 35%, Sharing/Dissemination – 33%, and Acquisition/Application – 32%). We analyzed the cases of the Capture/Creation phase using the states of knowledge [41]: tacit knowledge, which is in the stakeholders' minds; documented knowledge, which is codified in an informal/ad hoc manner, and formalized knowledge, which is codified in a standardized structure. We observed that only 7% of all the papers report a formalized means of coding AK, 11% report a documented means, 4% report a tacit means, and 13% do not specify how AK is captured. Most of the papers reporting a way in which to capture AK employ a volatile means to do so, since AK remains tacit or is informally codified. AK could, therefore, lose meaning over time or in another context, and there is consequently a lack of adequate means to capture AK in AGSD environments that ensure the duration of AK.

2.2. Architectural knowledge management solutions based on social tagging

As stated previously, this paper proposes the concept of AK Condensation (see Section 3), implemented using a prototype based on tagging personal interactions using UTEM in real time, as a means to classify AK so as to ease its subsequent retrieval. Researchers and software companies have chosen social tagging as a lightweight and unobtrusive manner to organize unstructured or dispersed data or to add meaning and metadata to software development environments, to recover knowledge that is generally hard to find. To the best of our knowledge, seven

[2] https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2.

Table 1
Tools that use social tagging in software development reported in literature (A=Analysis, D=Design, I=Implementation, T=Testing, M=Maintenance, Full=Cover all the phases). ArchiKCo refers to the prototype presented in this paper, which was designed for AGSD to cover any development phase and was focused on tagging UTEM interactions, where valuable AK is located. ArchiKCo tags are linked to base tags and include an auto-complete mechanism to ease the tagging action during the developers' interactions in order to avoid tag explosion. Finally, ArchiKCo has different parameters to perform AK retrieval, which other tools do not consider. See Section 3.2 for more ArchiKco details. *Parsed by TagSEA.

Tool name	Coverage			Tagging		Knowledge retrieval	Tool type
	Environment	Phases	Items to tag	Mechanism	Type		
IBM® Rational® Jazz® [44]	Agile distributed	Full	Artifacts and workitems	Auto-completed	Free	Free text and tags	Commercial
TagSEA [45]	Distributed	I	Sourcecode	Auto-completed	Free	Based on waypoints and tags	Research open source
Trac [69]	Distributed	I, T	Version control and Tickets (bugs)	Free	Free	Free text	Open source
eMoose [65]	Not defined	I	Source code	Auto-completed*	Free	Contextual "Push"	Research
CodeSnippets [70]	Not defined	I	Code Snippets in Source code	Free	Free	Based on tags	Open source
Paul et al. tool [71]	Open source	D, I	Software components	Free	Free	Free text associated with tags	Research
TAGGER [42]	Distributed	A	UTEM interactions	Free	Linked to base tags	Not reported	Research
ArchiKCo	Agile distributed	Full	UTEM interactions	Auto-completed	Linked to base tags	Free text, tags, dates, remittent, recipient, UTEM source	Research prototype

tools use social tagging (see Table 1). Most of these tagging tools are designed to support the implementation phase and are focused on tagging source code, software components or version control entries, i.e., they help manage AK. Only TAGGER [42] was designed to tag personal interactions in UTEM, but is focused on capturing domain knowledge during the analysis phase. Moreover, most of these tools are oriented toward a distributed development environment, and only IBM® Rational® Jazz® was evaluated in ASD. Our prototype, ArchiKCo (explained in Section 3.2), is shown in Table 1 in order to contrast its characteristics.

Regarding the implementation of tagging, only three tools have an auto-complete mechanism to aid during tag assignment. Moreover, most of them use free tags, i.e., there are no fixed or predefined tags to assign, and users are, therefore, free to write or compose any tag. Free tagging and unassisted tag assignment could lead to tagging difficulties and information retrieval problems caused by: (1) a huge number of tags, known as tag explosion; (2) differences in the interpretation of a tag's meaning; (3) an incomplete context in which to understand a tag; (4) the locality of tags, i.e., tags based on a team's jargon; (5) tags that only make sense when used together, known as composite tags, and (6) tags with the same meaning but written differently, known as obscure similarity [43].

Despite the above problems, literature reports that developers prefer using free tags because of their low cognitive load for everyday work [44,45]. Some efforts have been made to develop auto-tagging mechanisms [46] or tag-based recommender systems [43,47] to reduce developers' cognitive loads to an even greater extent. Sohan et al. [46] auto-tagging mechanism relates email messages to user stories in ASD projects with an accuracy of 70%; however, the remaining 30% of error could cause knowledge retrieval problems. Moreover, the tag recommender systems TagRec [47] and LS^{3}AutoTagger [43] are promising means to complement tag assignment and enhance the basic auto-complete mechanism.

Most of the knowledge retrieval mechanisms shown in Table 1 are based on tags and/or free text, except eMoose, which "pushes" AK to the users depending on the coding context. No tools except ArchiKCo base their knowledge retrieval mechanisms on dates, people and origins (i.e., a ticket, source code, or any other artifact). This is relevant, since agile/global developers struggle to find AK because they do not usually remember who originally provided it, or where and when a certain piece of AK was posted [19].

2.3. Architectural knowledge vaporization consequences in agile and global software development

The major challenges in AGSD are related to communication, culture, trusted relationships and KM [1,48]. A key success factor in any software development project is the correct appliance of KM [49], and consequently of AKM. As stated in Section 2.1, AKM is still a challenge in AGSD because most AK remains tacit or documented and could, therefore, lose meaning over time or in another context, i.e., it is prone to vaporization. It could be argued that AK vaporization in ASD is mitigated by practicing shared source code ownership [50], thus making developers aware of the project's AK. However, the four distances inherent in GSD cause inefficient AK sharing, since there are less opportunities for casual interaction [51] and informal awareness [52]. Shared source code ownership does not, therefore, have the same effect in AGSD.

AK vaporization could cause the following problems in AGSD [53,54]: (1) poorly understood requirements and technical solutions; (2) a lack of knowledge transfer between teams; (3) defects in software evolution and maintenance, i.e., architectural technical debt [55]; (4) a lack of visibility in project monitoring, and (5) time wasted by experts answering the same questions on certain issues and attempting to find solutions to problems that have already been solved. A consequence of the last point is that team members could annoy experts: constant questions could lead to an erosion in interpersonal relationships, which could affect the knowledge flow [56]. Interpersonal relationship erosion could be critical when building trusting relationships, which is important for any agile team. All the aforementioned problems and situations show the importance of addressing AK vaporization in AGSD. Our approach to mitigate this phenomenon is presented as follows.

3. A proposal of architectural knowledge condensation

As stated above, AK vaporization hinders the KM cycle because there is documentation debt. In a previous study [19], we found that UTEM logs contain valuable documented AK, and that developers attempt to recover it from those sources. Finding AK in UTEM logs is difficult because these media are not designed to find knowledge and because AK storage is unstructured. Moreover, it is difficult to retrieve AK because it is dispersed among different UTEM, which developers use to communicate. In this section, we present our contribution to AKM in AGSD, proposing a means of structuring and retrieving the AK shared using

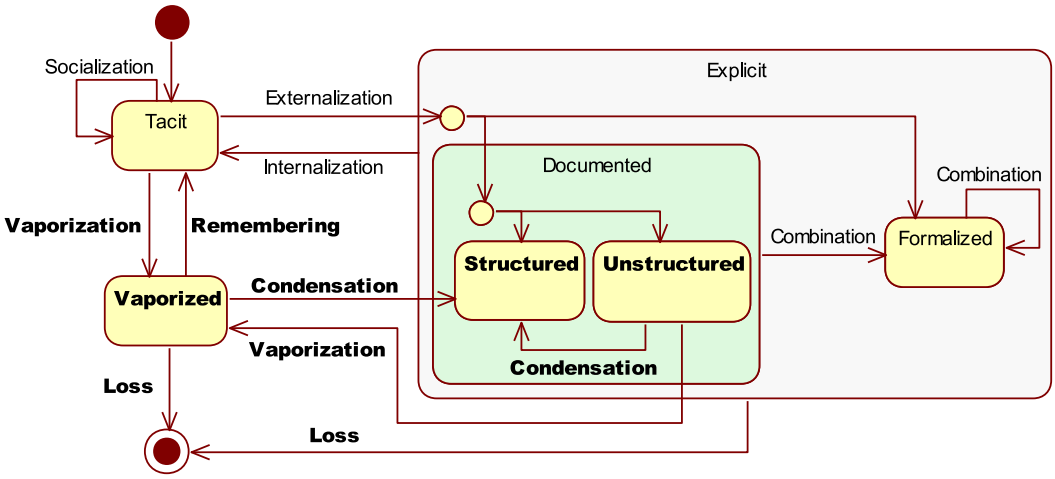

Fig. 1. UML state diagram representing SECI model with documented and formalized sub-states, and extended with the vaporization and condensation concepts. States and transitions in bold are product of our SECI model extension.

UTEM, in which AK structuring is based on a lightweight classification mechanism. This proposal is called AK Condensation – the opposite of AK vaporization. We additionally present a prototype that implements AK Condensation in order to evaluate the proposal's feasibility.

3.1. Conceptual definition

Explaining the AK Condensation concept implies exploring the vaporization concept in greater depth. Various authors define AK vaporization as the disappearance of AK owing to documentation debt [22,57,58]. However, we propose that vaporization could be a state just before AK disappears. In AGSD, developers attempt to find AK in UTEM logs when AK has evaporated from their minds. AK is, therefore, still recoverable because UTEM logs contain AK traces [19], which could help them infer/remember AK. In order to show our concept of AK vaporization and condensation, we extend the SECI model [41]. Fig. 1 shows the AK vaporized state, which may occur when developers forget tacit AK or cannot find AK in any kind of unstructured repository (e.g. UTEM logs). We propose that vaporized AK can be recovered when teammates help developers remember a piece of AK, or when unstructured AK or vaporized AK are gathered and structured to ease their retrieval (AK Condensation). Fig. 1 also shows that condensation is not a means to convert vaporized AK into AK in formal notation, but simply a step forward to ease AK formalization and reduce AK loss; we consider there still is a big gap between documented AK in UTEM and formalized AK. AK Condensation could, therefore, be implemented by considering the following elements:

1. **Accessible UTEM logs**. All stakeholders must be able to access all information from UTEM logs and thus be able to access all the AK being shared among them.
2. **UTEM log classification mechanism**. There must be a classification mechanism to structure the UTEM log information in order to ease AK retrieval. UTEM include features to find information in their logs. However, these features do not find AK efficiently [19]. In addition, developers do not usually remember the exact terms/concepts in which AK was shared and consequently need a semantic structure associated with the UTEM log to help them find AK without knowing the exact term to search for. In addition, this semantic structure would ease the transition from documented AK and formalized AK in later stages.
3. **AK searching mechanism**. All stakeholders could use the classification scheme to find valuable AK with less effort in the structured UTEM logs. The searching mechanism could include any other search parameter to ease AK retrieval, e.g., date period, message sender or message author.

Since AK Condensation is an abstract definition, there could be different ways to develop concrete instances. The following sub-section explains how we implemented a technological solution based on this concept.

3.2. Prototype of architectural knowledge condenser

In order to prove the feasibility of AK Condensation, we instantiated this concept using a technological solution called ArchiKCo, evaluated by professional developers and students (see Section 4). We based the ArchiKCo classification mechanism on social tagging, which can be applied during UTEM interactions. Social tagging is a lightweight and popular means to classify knowledge, which has been successfully used by other authors (see Section 2.2). Furthermore, in [59] we observed that social tagging is not a great effort for agile/distributed developers, and that they are interested in tagging UTEM messages in order to retrieve AK in the future.

We based ArchiKCo on Windows and Skype[3] (as the UTEM log source), since most of the subjects that were able to evaluate it use them both in their daily work. Fig. 2 depicts the ArchiKCo operation, showing the activities that implement the three elements of AK Condensation, along with the common situations described by agile/global developers (depicted as dialog clouds in Fig. 2); we explain how ArchiKCo implements each element below.

3.2.1. Accessible UTEM log information

We implemented this part using a Gatherer Service (see Fig. 2, part A) to periodically extract and send the Skype interaction logs to a shared repository in the cloud (depicted as a UTEM Messages database in Fig. 2). We used Algolia[4] server as a shared repository, since it provides robust indexing functions that ease the development of a searcher.

3.2.2. UTEM log classification mechanism

In order to avoid the problems related to free tagging [43], we implemented a semi-fixed tagging mechanism (successfully evaluated in [59]) that allows developers to add user tags with a web application called Tags Administrator (see Fig. 2, part B). We propose that developers perform this activity during a development cycle planning meeting,

[3] https://www.skype.com/.
[4] https://www.algolia.com/.

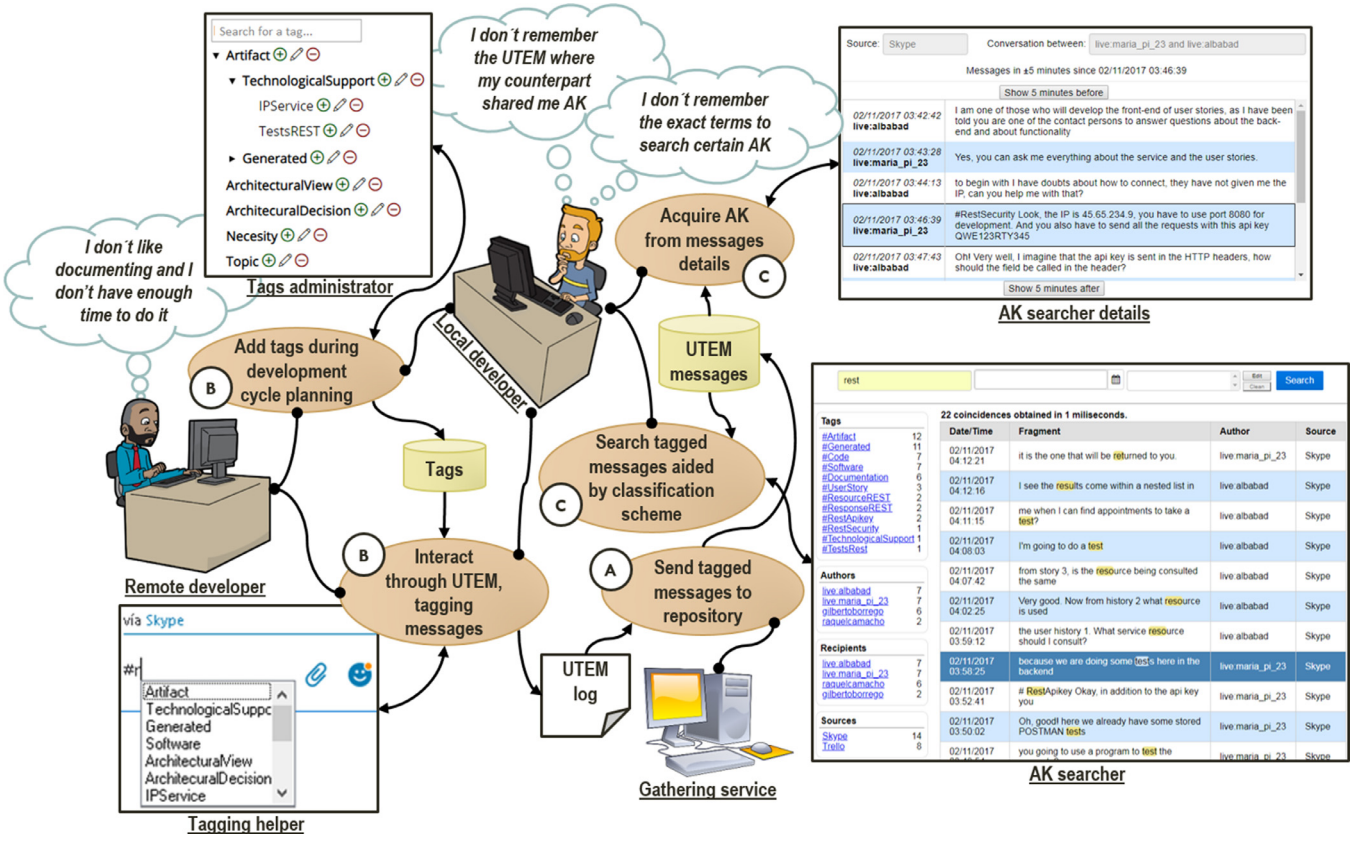

Fig. 2. ArchiKCo rich picture with activities (ovals) corresponding to the three elements of AK Condensation concept, A = Accessible UTEM logs information, B = UTEM logs Classification mechanism, C = AK searching mechanism. Bulleted lines represent links between activities and who performs them. Arrowed lines represent links between activities and artifacts. There are three types of artifacts: resulting artifacts (activities' outgoing arrows), source artifacts (activities' ingoing arrows), and interacting artifacts (linked with double arrow lines).

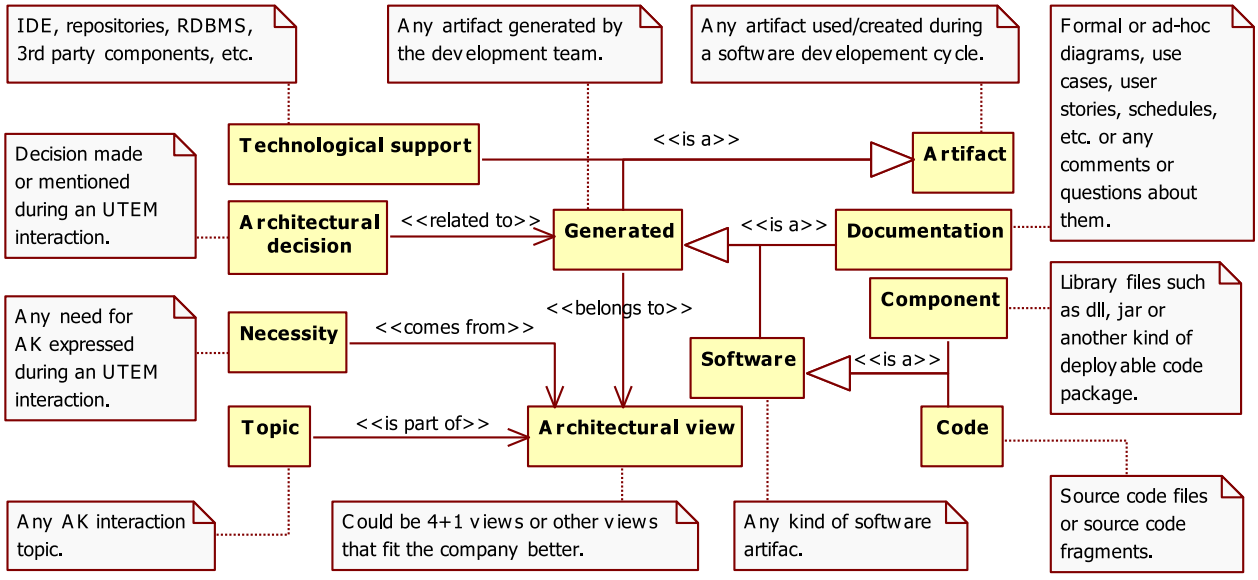

Fig. 3. Conceptual model based on UML, representing the aspects involved in AK articulation through use of UTEM by AGSD teams (reproduced from [19]).

since they would already know the key terms to be used during the next cycle. Every custom tag must be related to a meta-tag from an AK Model, which represents the AK that developers share; this model was empirically obtained in [19] (see Fig. 3). The aim of relating user tags and meta-tags is to provide an abstract means to store and find AK, signifying that when developers wish to recover AK and they do not remember the exact name of a tag, they can use a meta-tag to conduct an initial search.

Once the development cycle has started, remote and local developers could interact using UTEM, tagging messages that contain important AK. We are aware that developers sometimes make typing errors, or forget the exact way in which each tag was registered, or even the existence of certain tags, and have, therefore, developed a tagging helper component (see Fig. 2, part B) that auto-completes tags while developers are typing in conversations, whose source was the tag repository (depicted as a

Tags database in Fig. 2), which was updated using Tags Administrator. We thus aim to reduce the number of typing errors, ensure that users are using the exact tag writing and reduce tagging problems [43].

3.2.3. Architectural knowledge searching mechanism

We developed a web-based searcher, called AK searcher (see Fig. 2, part C), which has three search parameters: (1) free text, in which users can input any text to be searched for in the message content, along with the names of the author and recipient of the message; (2) Date range, from which users can select the period when the knowledge was shared; and (3) Tag filter, from which users can select the tags from a tag tree, on which meta-tags and user tags are hierarchically organized. This enables users to remember which tags are available and then add any number of them to the tag filter.

Having executed a search, a list of the coincident messages is shown, along with four panels (see Fig. 2, part C), which show all the related authors, recipients, sources (UTEM from which the messages originated) and tags of the resulting messages. Developers could apply extra filters to narrow the results, by clicking onto the panels' elements. The tag panel also shows each tag's parents; for example, if there is a tag called #nodeMongoDB, which is related to the #Code meta-tag, the tag panel shows both. We are aware that obtaining AK from a single tagged message could be difficult. AK searcher includes a feature to obtain the interaction context of a selected message, in which developers can read messages that were sent five minutes before and five minutes after a certain message (see Fig. 2, part C); they can even load messages from an additional five minutes before or after, if necessary.

3.3. Architectural knowledge condensation in agile and global software development

Instantiating the AK Condensation concept would give agile/global developers a lightweight means to structure AK (with a low cognitive load) while they are interacting with UTEM (maintaining agility), and an easier means to retrieve dispersed AK from UTEM logs. Our proposal could consequently reduce AK vaporization in AGSD environments. However, before implementing AK Condensation in a real scenario, we must first determine its concept feasibility in a controlled environment. We determined this concept feasibility by observing two key phases: AK structuring and AK retrieval. The following research questions arose from the latter:

- **RQ1.** Is an assisted semi-fixed tagging mechanism suitable to structure AK and avoid tagging explosion during UTEM interactions in AGSD environments?
- **RQ2.** Is it better to search for AK using the ArchiKCo searcher than by directly using the AK sources (UTEM) in AGSD environments?
- **RQ2.1.** Is ArchiKCo searcher trustworthy as regards finding correct AK?
- **RQ2.2.** Do developers find the correct knowledge faster using the ArchiKCo searcher than directly using the AK sources (UTEM)?
- **RQ2.3.** Is the ArchiKCo searcher preferable when searching for AK rather than directly searching in the UTEM source in AGSD environments?

The method employed to determine the feasibility of the AK Condensation concept using the ArchiKCo prototype is presented below.

4. Method to evaluate architectural knowledge condensation feasibility

4.1. Scoping

As stated above, AK structuring and AK retrieval are the key phases of the AK Condensation concept. We, therefore, designed a two-part evaluation to determine the feasibility of the concept presented. We define these parts below.

In [59], we observed that social tagging could be a lightweight manner to structure AK, using a semi-fixed and assisted tagging mechanism. In that study, we added a tagging helper to a web-based messenger, developed ex-professo, thus giving us full control over the tagging environment, allowing us to ensure that participants only used registered tags. The ArchiKCo tagging helper has now been added to Skype, signifying that we do not have sufficient control to avoid unregistered tags. In order to answer RQ1, the first part of evaluating AK Condensation comprised an observation study focused on tagging behavior in terms of registered (valid) tags and obtaining participants' perceived usability of the tagging mechanism.

We believe AK Condensation could enhance AK searching in UTEM logs during AK retrieval. We, therefore, designed the second part as a quasi-experiment to compare participants' searching performance when using ArchiKCo and two UTEM: Skype and Trello (RQ2). This section presents the method employed to determine the feasibility of AK Condensation, which was structured following Wohlin et al. [60] specification.

The objective of the whole study is to *Analyze* the use of the implementation of the concept of AK Condensation, *for the purpose of* determining its feasibility *with respect* to tagging behavior and AK retrieval, *from the point of view of* professional developers and students, *in the context of* AGSD.

4.2. Planning

4.2.1. Context selection

We had two experimental contexts: industrial AGSD and academia (replica). The participants in the industrial context were professional developers from seven Mexican companies: four small[5] (<100 employees) and three medium[5] (100–999 employees). The participant companies develop software for diverse areas (transportation, health care, internet of things, administration in general, etc.), have worked in a distributed or global environment, and all work in an agile manner. The main objective of working in an industrial context was to attain richer qualitative feedback about the AK Condensation concept. The academia replica took place at Castilla-La Mancha University in Spain (Spanish acronym, UCLM) with undergraduate and graduate students from the Superior School of Computer Science, all of whom had knowledge of AGSD.

4.2.2. Selection of subjects

The subjects of both contexts were chosen for convenience. The academia subjects were 3rd year undergraduate students, who had already studied subjects regarding agile methods, programming and software design. There were also graduates researching topics related to software development, who consequently also know about agile methods and software design. The industry subjects were professionals who have worked on AGSD projects.

4.2.3. Study design

This study had a within-subject design, since all the treatments were applied to all the participants. It consisted of two parts: AK structuring and AK retrieval. The first part was an observational study during which all the participants were mentally situated in a context scenario to interact in pairs using Skype and the ArchiKCo tagging helper (with predefined user tags) and following a chatting script containing seven marks suggesting what to tag. Since the participants did not register the user tags in the catalog, they were free to assign unregistered tags if they did not find one that fitted a certain message.

In the AK retrieval part, the participants had to answer a 12-question survey concerning the context scenario in an attempt to emulate AK needs. The survey answers were stored in the Skype log generated in the previous part, and on a Trello board that contains user stories and

[5] https://www.gartner.com/it-glossary/smbs-small-and-midsize-businesses/.

comments related to the same context scenario. We chose Trello as a second UTEM because it is easy to use and commonly used by the participants. Eight survey questions indicated which media to use to search for the answer: Skype, Trello or AK Searcher, which contains the logs of both UTEM. The participants were free to choose the media they preferred to search for answers to the last four questions, which were designed so that the answers to questions 9 and 10 could be found using Skype or AK Searcher, and the answers to questions 11 and 12 could be found using Trello or AK Searcher. We consider that this part was a crossover quasi-experiment with one factor and three treatments, in which only two comparisons are relevant: AK Searcher/Skype and AK Searcher/Trello.

4.2.4. Variables selection

In the AK retrieval part, the independent variable was represented by the different media used to search for AK. There is no independent variable for the AK structuring part, since it is an observational study. The dependent variables were: tag validity, i.e., number of registered and unregistered tag instances used by participants during the chatting session; media preference, i.e., percentage of time a participant used a certain media to search for AK; correctness of answers, percentage of correct AK found per media; and time required to find correct knowledge.

4.2.5. Hypotheses formulation

In this part, we present the four study hypotheses, which are directly related to the dependent variables defined above.

- $\mathbf{H_{0TagsValidity}}$: There is no difference between the number of valid and invalid tag instances used during the UTEM interaction.
- $\mathbf{H_{0Preference}}$: There is no difference as regards to the preference to search for knowledge using any of the media provided.
- $\mathbf{H_{0Correcteness}}$: There is no difference in the percentage of correct answers found using any of the media provided.
- $\mathbf{H_{0Time}}$: There is no difference in the time required to find correct answers using any of the media provided.

4.2.6. Instrumentation

Below, we present the instruments developed in order to conduct this study as it was designed.

- **Context scenario**. This scenario concerned two agile developers from different companies and locations working on the same project (medical appointments system), one of whom required information about a RESTful service that the other was developing. They had documentation debt, and consequently had to acquire the project AK by asking each other questions.
- **Chatting scripts**. Each pair of participants had to follow 2 scripts (one per scenario role) to simulate a technical conversation taking place using Skype regarding the context scenario.
- **SUS questionnaire**. We prepared a questionnaire based on the System Usability Scale [61] (SUS)using a Likert-7 scale and focused on the Tagging Helper. We added two SUS-style questions (one positive and the other negative) to explore the participants' perceptions of the helper's unobtrusiveness. This questionnaire also included an open-ended question to request suggestions regarding the Tagging Helper.
- **Extended TAM questionnaire**. We prepared a questionnaire based on the Technology Acceptance Model [62] (TAM) using a Likert-7 scale and focused on the AK Searcher. We added questions concerning reductions in interruptions (one question), finding relevant AK easily and in a timely manner (three questions) and the participants' overall impression of the whole ArchiKCo prototype (two questions).
- **12-questions survey**. This survey was uploaded onto LimeSurvey.[6] To compare the participants´ performance using AK Searcher versus Skype and Trello, we created two survey versions (one for each pair member), in which we varied the indicated media per question. For instance, while one pair member was required to answer a question using Trello, the other pair member was required to answer the same question using AK Searcher; and the same between Skype and AK Searcher.

4.3. Operation

4.3.1. Preparation

We deployed an ArchiKCo instance for each pair of participants and registered 10 user tags linked to different meta-tags related to the context scenario. The user tags were: IPService, TestsREST (related to TechnologicalSupport meta-tag); RestApikey, RestSecurity, Encryption, TestData, RESTResource, RESTResponse (related to Code meta-tag); AngularEncryption (related to Component meta-tag); and UserStory (related to Documentation). We pre-defined tags because each participant pair could define a different set of tags, which hindered the tagging behavior analysis. We additionally carried out a pilot test, which showed that the pre-defined tags really accorded with the context scenario. We also added five cards to a Trello public board on which fictitious members of the development team provided user story clarifications. Finally, we activated the two versions of the 12-question survey.

4.3.2. Execution

We first carried out the study in the industrial context with 30 professionals (average age = 28, SD = 3.9), 10 from the medium-sized companies, and the rest from the small ones. The industry participants had experience in ASD (average of 3.1 years' experience, SD = 1.9) and in GSD (average of 2 years' experience, SD = 1.4). The study in UCLM was carried out three months after the industry study, with 30 students (average age = 24.1, SD = 3.5): four graduates and 26 undergraduates. The experiment took place in three sessions per week for both contexts, so not all participant began on the same day. These sessions are explained below.

- **Installation session** (duration ≈ 10 min). The participants were given an overall explanation of the study sessions along with their objectives. We organized the participants into pairs and then helped them configure the tagging helper (to work with Skype) and the Skype extractor (to send each pair's conversations to a shared server).
- **Solving scenario session** (duration ≈ 25 min). We gave the participants a short training session regarding how to use the tagging helper (three minutes, approx.), and they quickly explored the available tags (two minutes, approx.). We then described the scenario in which they would be located to carry out the tasks, and assigned a role to each pair member: either the developer working on the RESTful service or the developer who wished to use it. Each member of each pair sat in a different part of the session room, ensuring they had no visual contact, as if they were geographically distributed. We asked them to avoid talking to each other to better emulate an environment of geographic distribution. They then used Skype to chat, following the corresponding script, and tagging aided by the Tagging helper. We also told them that they could tag any message as they considered necessary, and that they could write a new tag (unregistered/invalid tag) if they could not find one that fitted a certain message on the options shown by Tagging helper. After the participants had finished following the chat script, they answered the SUS-based questionnaire.

[6] https://www.limesurvey.org/.

- **Searching session** (duration ≈ 20 min). This session took place two days after the chatting session to prevent the participants from being able to remember the chat topics, thus mitigating the learning effect. The participants were required to answer the 12-question survey easily. The version of the electronic survey was assigned to each pair member randomly. The participants were trained to search in the three different media, after which we explained that each question indicated where to search for the answer, along with the fact that they could answer "I don't know" if they were unable to find any information. They then responded to the corresponding electronic survey and finally to the extended TAM questionnaire.

4.3.3. Data collection

The tags' validity was determined by obtaining all the tagged messages and comparing them with the tags catalog (user tags and meta-tags) to obtain the number of valid and invalid tags per participant. Regarding media preference, each 12-question survey had a field for the last four questions in which the participants indicated the media used to obtain the answer. We, therefore, counted only the number of answers for each media. Correctness of answers was obtaining by manually checking each one. The time required to find the correct knowledge was measured using LimeSurvey, which registers the time that has elapsed between the presentation of a question and that at which the participant clicks onto the next button to pass to the next question. Finally, we obtained the qualitative perception about Tagging Helper and AK Searcher using the results of the SUS and TAM questionnaires, respectively.

4.3.4. Data validation

The AK retrieval part of the industry context was conducted in the respective participant companies. We could not avoid interruptions during the session and the time required to find the correct knowledge was consequently affected. We, therefore, take into account only the time data from the academic context.

5. Results

The results obtained are presented in three parts: (1) the results of the AK retrieval part including the Tagging helper usability perception; (2) those of the AK retrieval part including the participants' perceptions of AK Searcher; and (3) the participants' overall perceptions of ArchiKCo.

5.1. AK structuring part

In this section, we analyze how the participants tagged messages using the tagging helper and its usability perception. During the tag analysis, we also identified tag instances that were used correctly in terms of their semantics. The results are consequently presented in terms of tag validity, tag correctness and Tagging Helper usability and unobtrusiveness.

5.1.1. Tags validity

Fig. 4 shows that most of the professionals (80% approx.) used between six and nine tag instances during the chatting session, while the UCLM students (80% approx.) used between three and eight tag instances. A considerable percentage of the participants (50% approx.), therefore, used valid tags as required, or more, i.e., at least seven tagged messages. Moreover, some participants used 12 or more valid tag instances (13% approx.), i.e., at least five instances more than expected. We can, therefore, interpret that they had the initiative to tag messages when this was not suggested in the script.

Upon considering invalid tag instances, 23% of the professionals did not use invalid tags, while only 6% of UCLM students did not do so (see Fig. 4). Furthermore, around 73% of the professionals used between one and three invalid tag instances, while 83% of the UCLM students used between one and six invalid tag instances. Invalid tags also represent new tags, and in this respect, 53% of the UCLM students used between one and three instances of new tags, while only 23% of the professionals used one new tag instance (see Fig. 4). However, 33% of the UCLM students used between four and eight new tag instances, indicating their disposition to tag messages or their inexperience with the script topics, signifying that they had to create new tags that would fit their knowledge. Fig. 4 also shows that around 50% of all the participants made no typing errors in the tag instances, and that 46% of the participants had only one or two "typos". This could mean that the Tagging Helper really helped them obtain a low error rate.

5.1.2. Tagging correctness

Around 13% of all the participants had no semantic errors when using tag instances, and 40% used only one or two tag instances erroneously (see Fig. 4). This is significant, because the participants did not know the exact semantics of the tags beforehand. However, 50% of the professionals had between three and five instances of incorrect use, while only 23% of the UCLM students did so in the same range. Regarding the correctly used tag instances, 78% of all the participants had between two and six correct instances, and the professionals registered more variability than the UCLM students (see Fig. 4). Finally, around 13% of the participants registered between eight and ten correctly used instances, again highlighting that they did not know the tags in advance.

5.1.3. Overall tagging behavior

To summarize the tagging behavior (see Fig. 5), the participants used more valid tag instances (75%) than invalid ones (25%), and more tag instances were used correctly (47%) than incorrectly (28%). Both differences were confirmed statistically using the Wilcoxon Signed-Rank Test ($\alpha = 0.05$): valid vs. invalid instances – p-value = 0; correct vs. incorrect instances – p-value = 0.000008. Moreover, there were 18% of new tag instances and only 7% of typing errors, which was also statistically significant using the same test (p-value = 0.0056). This signifies that the invalid instances resulted more from the need for new tags than from errors caused by the tagging mechanism.

We also noticed that 38% of the correct tag instances were unexpected (see Fig. 5), i.e., tag instances that fitted the message's semantics but that the participants were not expected to use, or that instances were even used in messages in which we did not suggest tagging. Unexpected tags comprise 26% of user tags and 12% of meta-tags. These meta-tags were: Technological Support, Component, Software, Code, and Necessity. These meta-tags would, therefore, appear to be intuitive, since we did not explain their meaning.

5.1.4. Tagging helper usability and unobtrusiveness

We obtained the scores from all the SUS questionnaires and transformed them according to the curved grading scale [63], including the two questions regarding unobtrusiveness. Tagging helper obtained an averaged SUS score of 77 (SD = 13, med. = 78 = B+), corresponding to a B grade, and represents a good usability perception [64]. Both sets of participants had similar usability perceptions. However, around 20% of the professionals had lower usability perceptions (grades D and F). This was mainly owing to the mechanism employed to select a tag and navigate through the suggestion list; it was necessary to press the <Alt> + arrows to navigate, and the <Alt> + <Enter> to select a tag. Moreover, both sets of participants had similar unobtrusiveness perceptions; we obtained an average unobtrusiveness score of 72 (SD = 20, med. = 83 = A), which corresponds to a C+ grade. In this case, 30% of the UCLM students and 17% of the professionals considered that Tagging Helper would be obtrusive in their daily work. Upon analyzing the participants' comments, we concluded that this was caused by the navigation and selection mechanism and also by the low availability of tags, since the participants did not create/register the tags.

In summary, we observed a significant rate of valid tag instances used correctly during the chat session, with a low rate of typing errors. This behavior could lead to a reduction in tag explosion and its

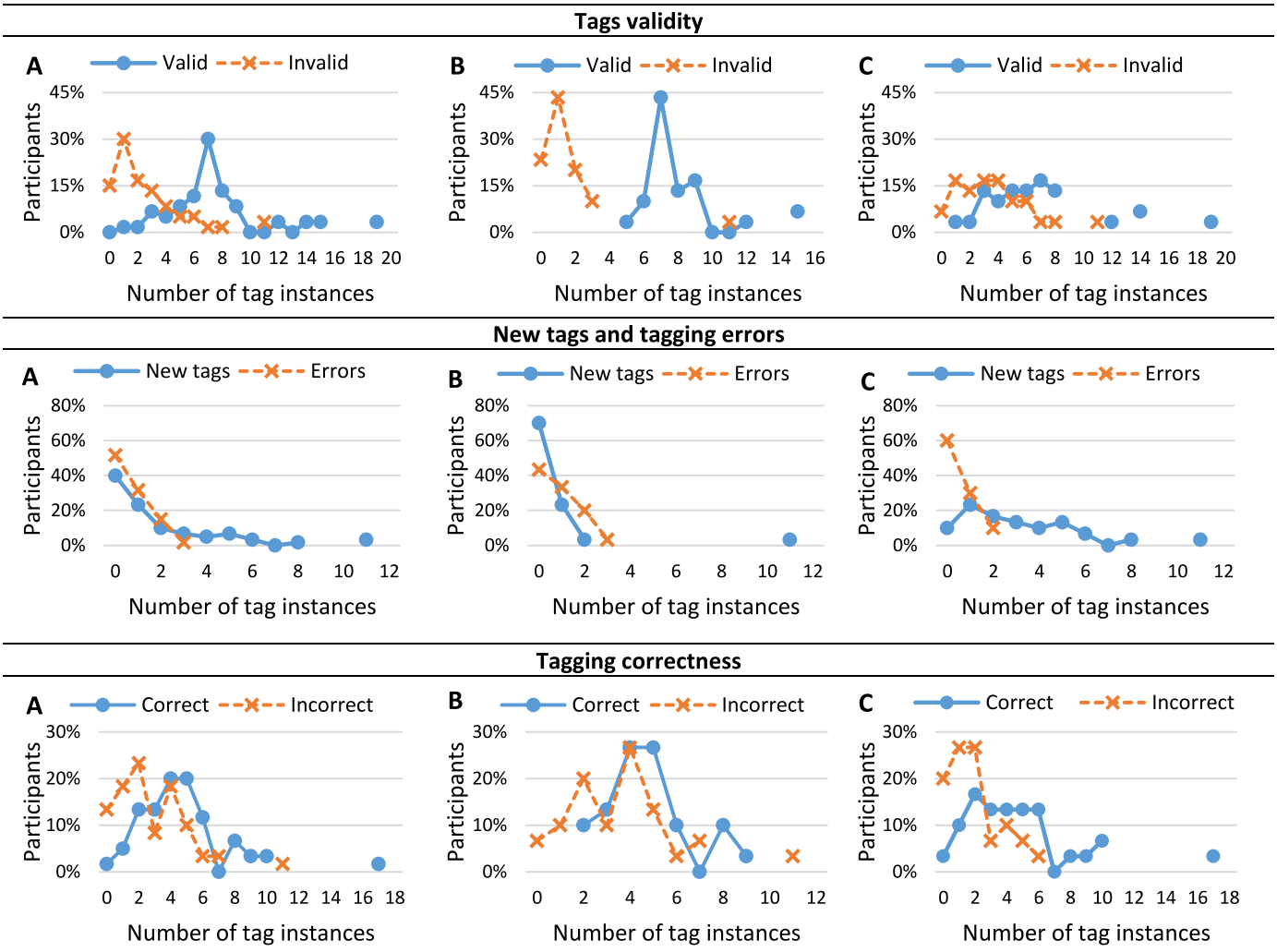

Fig. 4. Tagging behavior expressed by tag instances per participant.A = Mexican developers and UCLM students, B = Mexican developers only, C = UCLM students only. Outliers correspond to identified participants (< 2) who used many valid/invalid tag instances, or many new tags instances, or many correct/incorrect tag instances.

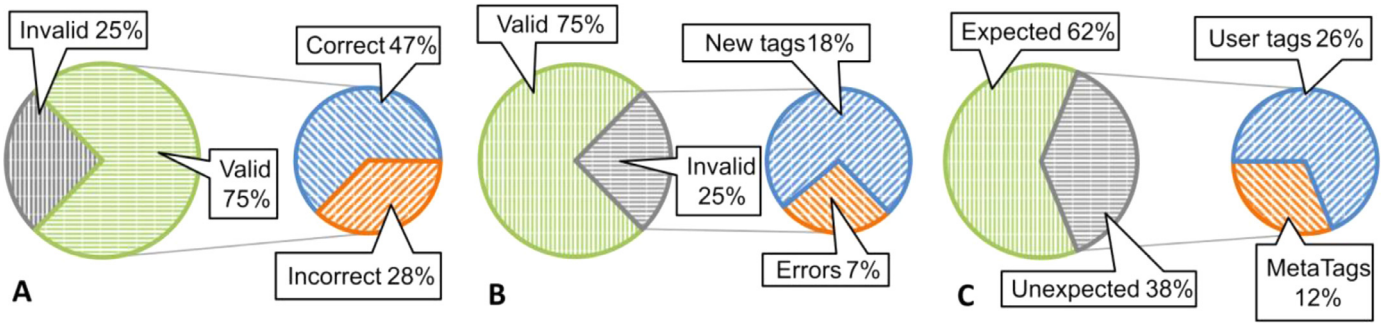

Fig. 5. A = Distribution of all tag instances (valid and invalid), detailing valid tags. B = Distribution of all tag instances (valid and invalid), detailing invalid tags. C = Distribution of correct tag instances (expected and unexpected), detailing unexpected.

related problems. Moreover, Tagging Helper has a great chance of being adopted to structure AK, since it is perceived as usable and unobtrusive.

5.2. AK retrieval

5.2.1. Media preference results

The participants preferred AK Searcher to Skype and Trello when answering all the questions (see Fig. 6, part A). There were questions to which the participants could not find the answers and did not, therefore, register a preferred media. Focusing only on the answered questions, and considering both participant profiles (professionals and students), AK Searcher and Skype obtained preferences of 69% and 31%, respectively, by joining the preferences attained for questions 9 and 10. AK Searcher and Trello similarly obtained preferences of 71% and 29%, respectively, when joining the preferences attained for questions 11 and 12. Since using the data obtained for each question was insufficient for the use of a paired test such as the Wilcoxon Signed-Rank Test, we applied goodness of fit tests based

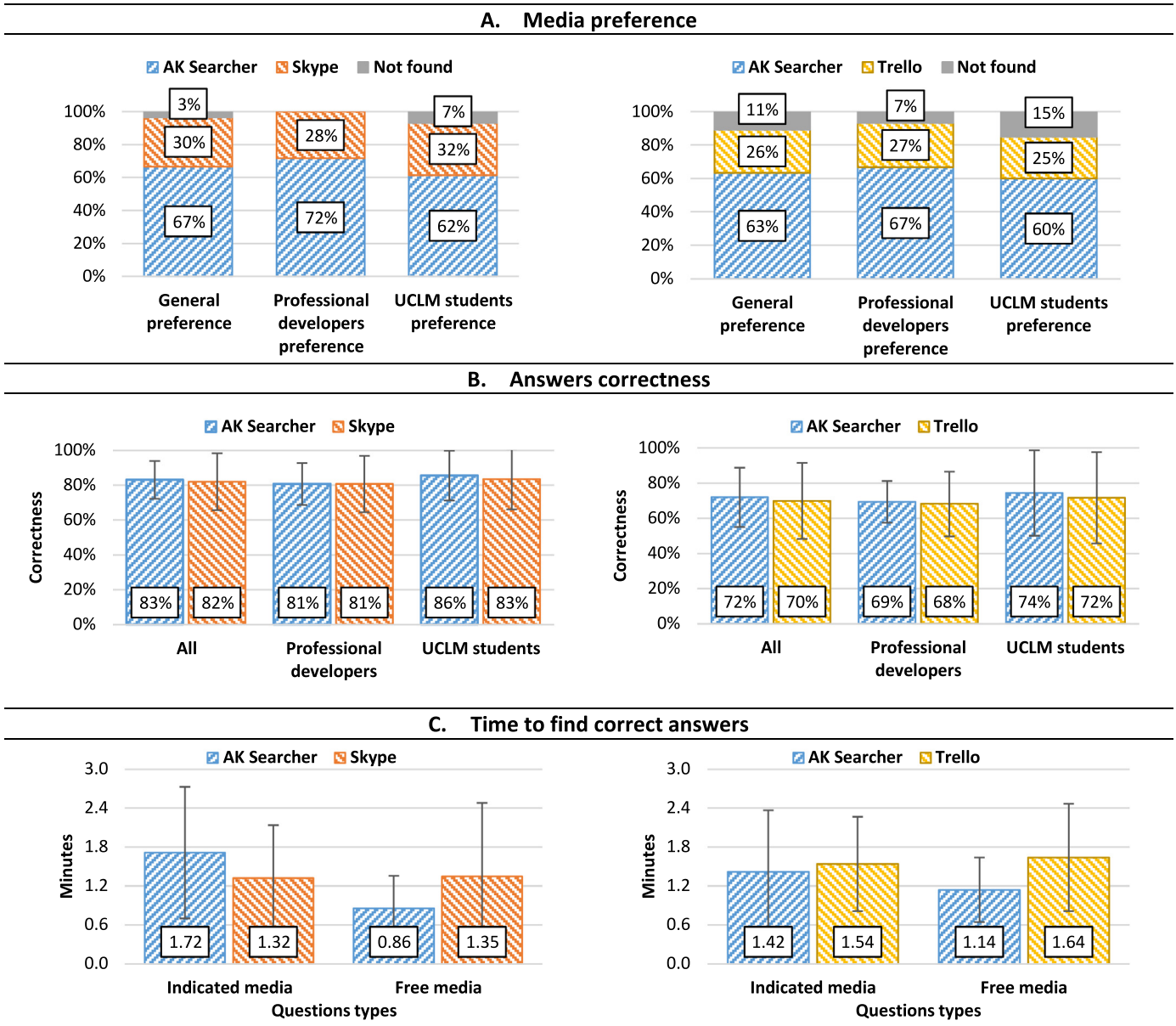

Fig. 6. A = Media preference of the participants as regards questions in which they had free choice. B = Comparison between correctness of answeres obtained using AK Searcher and Skype. C= Answering average times per question type. Vertical lines in B and C represent standard deviation.

on $X^2(\alpha = 0.05)$, supposing a uniform distribution for media preference. There is sufficient evidence to state that the participants' preferences are not uniformly distributed (AK Searcher/Skype questions: $X^2 = 16.69$, p-value < 0.001; AK Searcher/Trello questions: $X^2 = 18.925$, p-value < 0.001) and that there is, therefore, a tendency to prefer AK Searcher in all cases. We can, therefore, reject the null hypothesis $\mathbf{H_{0Preference}}$ since there was a significant difference in the preferred media.

5.2.2. Correctness of the results

Fig. 6 (part B) shows there were cases in which participants using AK Searcher obtained a higher correctness of answers than when using Skype, while there were others in which they obtained a lower correctness of answers than when using Skype, and yet others in which the correctness of the answers was the same for both media. Furthermore, both sets of participants behaved in a similar way when using either AK Searcher or Trello (see Fig. 6, part B). However, in this case, AK Searcher appears to have obtained a higher correctness of answers in more cases. We grouped the correctness data regarding questions 1–4, 9 and 10 (AK Searcher vs Skype), and questions 5–8, 11 and 12 (AK Searcher vs. Trello) of both participants' profiles to obtain two paired sets of data (AK Searcher vs. Skype and AK Searcher vs. Trello), each of which contained 12 elements, i.e., the results obtained for six of the students' questions and six of the professionals' questions. We then applied the Wilcoxon Signed-Rank Test ($\alpha = 0.05$) to both sets, and obtained that there is no difference among the correctness of answers when using AK Searcher or Trello ($W = 15 > W_{\alpha=0.05} = 3$), or using AK Searcher or Skype ($W = 30 > W_{\alpha=0.05} = 10$). It is not, therefore, possible to reject the null hypothesis $\mathbf{H_{0Correcteness}}$.

5.2.3. Time to find correct answers

Fig. 6 (part C) shows the time required to obtain answers with AK Searcher was higher than that required when using Skype for most of the questions in which the media required to search for the answer was indicated (questions 1–8). However, when the participants were free to choose any media (questions 9–12), less time was required to

obtain answers with AK Searcher than with Skype and Trello. We grouped the data by media and type of questions (free media choice or indicated media) such that we obtained four groups: (1) indicated media – Skype vs. AK Searcher, (2) indicated media – Trello vs. AK Searcher, (3) free media choice – Skype vs. AK Searcher, and (4) free media choice – Trello vs. AK Searcher. We determined whether there was a statistical difference within these groups by applying a Mann–Whitney U Test ($\alpha = 0.05$), because the samples are not paired, since we consider only the time required to find the correct answers. In the case of questions with an indicated media, there is a large difference in the answering time (p-value = 0.009) between AK Searcher and Skype, indicating that the participants found the correct answers faster when using Skype. In the same case, there is no considerable difference between AK Searcher and Trello (p-value = 0.153), although the participants were, on average, faster when using AK Searcher. In the case of questions with a free media choice, there is a large difference in the answering time (p-value = 0.045) between AK Searcher and Trello, indicating that the participants found the correct answers faster when using AK Searcher. However, there is no considerable difference between AK Searcher and Skype (p-value = 0.254), although the participants were, on average, faster when using AK Searcher. We can, therefore, reject the null hypothesis $\mathbf{H_{0Time}}$, since there were cases in which the participants were faster when using AK Searcher and others in which they were faster when using Skype.

5.2.4. Extended TAM results

The results of the extended TAM questionnaire showed that AK Searcher is extremely useful (median = mode = 6) and easy to use (median = mode = 6). The participants perceived that AK Searcher could help them find important AK in a timely manner (median = mode = 6). They also perceived that interruptions could be reduced using AK Searcher (median = mode = 6), since they would have a source of AK other than that of their teammates. It was also perceived that AK Searcher could ease the discovery of AK during development cycles (median = mode = 6).

5.2.5. AK retrieval results summary

AK Searcher was greatly preferred by the participants when searching for AK, which is supported by the perceptions presented above. Our results also provide evidence that professionals may perform better when locating knowledge if they use AK Searcher rather than Trello or Skype when they do not know the knowledge source. Finally, AK Searcher could be trustworthy when searching for knowledge, since the participants obtained a high percentage of correct answers and there was no significant difference between this percentage and that obtained with Skype or Trello.

5.2.6. Lessons learned by using Archikco prototype

We included a question in the TAM questionnaire to obtain a rating for the ArchiKCo prototype: a median of 8 (mode = 8) on a scale of 10. We were additionally able to learn some lessons from this study, which we grouped into AK structuring, AK retrieval and AK Condensation.

5.2.7. AK structuring

The participants stated that they have to get used to tagging their conversations. However, we believe that getting used to tagging could be easy because people are currently used to tagging in social networks. Furthermore, the participants suggested some enhancements to Tagging Helper: an automatic tagging or smart tag suggestion depending on the conversation topics, adding tags during the conversation, and selecting tags from a trending topic list. Finally, some participants struggled with the mechanism employed to browse tags in Tagging Helper, which was affected by Skype´s keyboard functions. This mechanism could, therefore, vary regarding the UTEM selected.

5.2.8. AK retrieval

Some participants commented that tags were not relevant during AK retrieval; one participant stated, "*I would end up not using labels, since I would look for words, not for labels*". In fact, while the participants were using AK Searcher, we observed that most of them preferred to search using free text, but some of them used tags to refine the results.

Regarding detail browsing interactions, this should change depending on the type of media: synchronous or asynchronous media; for instance, in asynchronous media, the time that elapses between messages could be more than five minutes, which is the time window that searchers have configured by default. We discuss this topic at greater length in the discussion section.

We also noticed that there should be a means to exclude informal messages from the repository. In this respect, one participant said AK Searcher "*could cause a lot of distraction because really important conversations are mixed with personal conversations, jokes, etc.*"; this situation could be problematic for AK retrieval.

5.2.9. AK condensation

Most of the participants commented that AK Condensation could speed up AK retrieval, reduce interruptions and reduce the repetition of information among teammates. The participants also appreciated that AK Searcher could be a single point of reference, rather than searching in multiple sources multiple times. In this respect, one participant stated: "*…notes and requirements are not discussed in the same place… there are conflicts because not all the parties have access to this information all the time.*"

6. Threats to validity

In order to understand to what extent the results are valid and how they can be used, a discussion regarding the validity threats is presented below according to Wohlin et al. [60] specification.

6.1. Conclusion validity

This study comprised participants from different backgrounds, but all of them were familiar with Skype and Trello. However, in order to balance the participants' knowledge, they received a brief amount of training regarding how to search for information in these media. We are aware that the participants could have been biased toward AK Searcher, because it was introduced as a new tool, or because they might have wanted to please us; however, their participation was anonymous. Moreover, we did not have previous contact with them before or after the evaluation and there was, therefore, no reason to try to please us. Moreover, the researchers were not close to the participants during the tasks in which they were free to choose one of the three available media. Furthermore, we are aware that the evaluation period was, perhaps, short. However, the results indicate an initial trend. It is also significant that, despite the short time and the use of a new tool, (people do not generally like to change their way of working) the subjects preferred AK Searcher when they did not know the location of a certain piece of knowledge.

6.2. Internal validity

All the participants were volunteers and showed a great interest in collaborating in this study. In addition, the study sessions were short to prevent them from getting bored or tired. We attempted to avoid learning effects by using a counterbalancing technique, i.e., we placed the participants in groups and presented the conditions (indicated media to search) to each group in a different order (see Section 4.3.2). Regarding persistence effects, the study was run with subjects who had never taken part in a similar study. Moreover, the participants did not have any previous knowledge of the context scenario, since it was fictitious. Furthermore, we conducted the searching session two days after the

chatting session to avoid the situation of the participants remembering all the details about the script topics. In order to clearly observe the results of the treatments on the participants' performances, they received the same set of questions to be searched for in the three media. All the questions could be answered using the indicated media, thus reducing the risk of the participants not being able to find the correct answer. The Wilcoxon Signed-Rank Test confirmed that there was no significant difference in the correct answers according to the media used, and the results are consequently independent of the study package.

6.3. Construct validity

We measured the time required to answer a question using a Limesurvey feature, which registers the time between a question being shown and the participant clicking onto the button to show the next one. This time could have been affected by the participants' reading speed and by the time needed to understand the question. We considered that the participants had similar abilities and this threat could, therefore, have been reduced by the arrangement of the sets. In order to discover the perceptions of Tagging Helper as regards unobtrusiveness, we extended the standard SUS questionnaire by adding two questions (one positive and the other negative), which were also processed by following the steps required to obtain the SUS score. This extension provided us with a structured means to obtain the participants' perceptions of topics that the conventional SUS questionnaire does not include.

6.4. External validity

We identified two main threats to external validity: subjects and tasks/materials. Regarding subjects, we also included students in order to have more controlled conditions in an academic context. Unfortunately, these students had no experience of real AGSD projects, but they had taken courses concerning ASD and GSD during their university education. We included professional developers with experience in AGSD to enforce external validity. Concerning tasks/materials, the chatting scripts were based on a fictitious scenario, but with real world characteristics. Although, tagging was suggested in the scripts, the participants also tagged messages using their own initiative. Moreover, the need for AK was motivated by a questionnaire, not by a project necessity. Real scenarios should, therefore, be considered, as they are supposedly more complex and articulated.

7. Discussion

In this paper, we propose the concept of AK Condensation and present its implementation (ArchiKCo), which was evaluated to determine the feasibility of this concept. These evaluation results are discussed in three parts: (1) AK classification mechanism, (2) AK searching mechanism, and (3) Feasibility of AK Condensation.

7.1. AK classification mechanism

Literature reports that developers prefer using free tags in tagging systems, given their low cognitive load in everyday work [44,45]. We based our AK classification mechanism on an assisted tagging mechanism (Tagging Helper), as IBM® Rational® Jazz® [44], TagSEA [45] and eMoose [65] do. However, we included predefined user tags, which are in turn associated with a fixed set of meta-tags, rather than just allowing free tagging. Our results do not reflect that the participants disliked using predefined tags, and they merely stated that it would have been better if they could have defined the tags that they used during the evaluation. However, we should explore the participants' perceptions in more long-term studies. The participants also stated that it might be appropriate to include a mechanism by which to add tags on the fly or a smarter tag suggestion (based on the context of the topics), but none of them mentioned free tagging.

The participants did not show any sign of disliking tag conversation messages and around 50% of them used the expected number of tag instances (seven instances). In a previous study [59], around 90% of the participants used at least the number of tag instances provided, although in that case, only three suggestions were marked in the scripts, which were also shorter. Both results show evidence that developers may need to tag messages during conversations, thus indicating that an AK classification mechanism based on social tagging could be successful.

We decided to use an assisted tagging mechanism to avoid problems such as tag explosion, which could ruin the AK classification mechanism. Our results do not show any signs of tag explosion. Despite the fact that we allowed the participants to use new tags if they considered it necessary, there were only 18% of instances of unregistered tags. This tagging mechanism could also help reduce the problem of obscure similarity [43], since the participants selected a tag from a suggestion list and messages were, therefore, tagged with correctly written tag instances. In that respect, there were only 7% of tag typing errors during the chatting sessions. This low error rate also contributed to keeping the AK classification mechanism functional.

Although the participants did not know the registered tags beforehand, significantly more tag instances were correctly (47%) rather than incorrectly used. This means the tags' semantics corresponded to the message topics. However, this tagging accuracy is lower than that obtained by Sohan et al. [46], who used an intelligent auto-tagging mechanism (70% accuracy). We believe that the accuracy of Tagging Helper could be increased in future evaluations if the participants define their own tags.

In our previous study [59], 30% of the participants used meta-tags correctly, while in the present study this percentage increased to 43%. This could, therefore, be considered as evidence that the conceptual model entities on which meta-tags are based are expressive and are related to AGSD-type situations in terms of AK. However, we should conduct studies focused only on the refinement of the model.

These results lead us to believe that Tagging Helper could be part of a good AK classification mechanism for use during UTEM conversations. Although the participants perceived Tagging Helper to be usable (SUS score 77 = B grade), this perception was lower than that attained in our previous study [59] (SUS score 87 = A+ grade = Excellent according to Bangor et al. [64]), in which we evaluated another implementation of Tagging Helper that was integrated into a custom Web instant messenger. In that study, we had absolute control over the autocomplete features and the participants, therefore, reported fewer problems with tag selection and navigation. However, the implementation employed in this study was integrated into Skype, signifying that we had to adapt the selection and navigation features so as not to interfere with the Skype features. Despite this difference in usability, the Skype-based Tagging Helper was perceived as unobtrusive (score 72 = C+ grade), but we believe that this perception could be improved if tag navigation and selection are also improved. Nevertheless, some participants stated that it was just a matter of getting used to the helper's features.

7.2. AK searching mechanism

Our results indicated that AK Searcher is a trustworthy application that is preferred by developers when searching for AK, because users tend to find AK faster than in UTEM when they do not know the knowledge source. However, AK Searcher was slower than Skype when we indicated a media in which to search. This could have been caused by the user interface design. While searching in Skype consists of at least three steps: (1) Ctrl+F to show the search textbox, (2) write text to search, (3) press <Enter>, AK Searcher requires two more steps to show a specific result: (1) write text to search, (2) press <Enter>, (3) look for an interesting result item, (4) open corresponding conversation, (5) look for knowledge in conversation. In a real situation, therefore, if a developer remembers the source of a certain item of knowledge, it might be better to search directly in that source. However, if a developer does not

remember the AK source and needs more than free text searching to find a specific item of knowledge, AK Searcher would be the best option, since it offers more parameters in which to carry out a search and refine the result set.

During the searching session, we observed that participants barely used the proposed classification mechanism to find knowledge, although they commented that tagging conversations is an interesting way to search for knowledge later. We believe that tags are more useful to search for AK if a developer wishes to attain knowledge from a general view to a detailed view. For example, when a new team member needs to acquire AK of the team's project, a good starting point might be to explore the existing tags and then explore the comments of a particular tag in greater depth.

The participants' comments and our observations led us to realize that improvements should be made to AK Searcher, one of which is to present the results differently depending on whether the AK source is a synchronous or an asynchronous UTEM. In our study, Trello could be considered as asynchronous, since card comments are not received as frequently as Skype messages in a conversation. The AK Searcher feature used to show a range of messages 5 min before and after a selected message may, therefore, be useless, since Trello comments could have a greater time difference. The same problem would occur with other asynchronous media (e.g. email). Another improvement would be to include a feature to show only tagged messages in the first result set. The problem of showing irrelevant messages (e.g. personal interactions, jokes, etc.) could, therefore, be reduced because these kinds of messages would not be tagged. Another way to reduce this problem would be to add exclusion tags that tell the Gatherer Service not to send certain messages to the repository.

To the best of our knowledge, there is no similar AKM research tool to ArchiKCo. The important differences between the reported tools (see Section 2.2) and our approach are: (1) they are not focused on searching for AK sourced in electronic interactions; (2) almost all of them are based solely on free text searching, and only TagSEA [45] and IBM® Rational® Jazz® [44] include extra parameters(e.g. tags or waypoints); (3) they do not have features to refine a result set; (4) they do not integrate AK from different sources (IBM® Rational® Jazz® could be configured to do so, but the paper [44] that refers to this does not present any integration), and (5) they do not present empirical results regarding searching in their respective papers. All these differences prevent us from making a direct comparison with our results.

Despite the improvement opportunities, AK Searcher was perceived to be a usable and useful media that eases the discovery of AK during an AGSD cycle, which could reduce interruptions among teammates when they have questions about the project architecture. Moreover, the participants perceived that AK Searcher would allow them to find AK in a timely manner, as required in an agile environment.

7.3. Feasibility of AK condensation

Agile/global developers know that UTEM logs contain important AK and they, therefore, need to search for architectural topics in those logs. However, they often spend too much time searching for AK because it is dispersed throughout different UTEM. The results obtained provide sufficient evidence to state that it could be feasible to implement the AK Condensation concept in AGSD for the following reasons.

- UTEM logs lack a structure that eases AK searching. We have, therefore, proposed a classification mechanism based on assisted social tagging. The participants used this mechanism well and it was perceived to be useful and unobtrusive. The results showed that agile/global developers could tag UTEM interactions accurately, even if they do not know the tags' meaning beforehand. In a real situation, developers should be careful to define useful tags, and to tag in a correct manner, since they are the most interested in retrieving AK from UTEM logs, because documentation debt is often present in AGSD environments. Furthermore, when developers tag during UTEM interactions, they are able to tag coherently because they are aware of the interaction topic. The evaluation results, therefore, allow us to state that (**RQ1**) social tagging is suitable and feasible to classify AK in AGSD because: (1) it is usable and unobtrusive, (2) agile/global developers could classify AK correctly, and (3) it is integrated into the agile work style.
- The AK retrieval mechanism was well received by the participants, since it integrates different AK sources, and they thus preferred searching for AK using this mechanism than searching directly in each UTEM. What is more, the participants tended to discover AK faster than when doing so directly in UTEM, particularly when they did not know which UTEM contained the AK required. The participants perceived that the AK retrieval mechanism was useful and usable. Furthermore, since AK is previously structured by the classification mechanism, AK retrieval could be easier and quicker than code analysis, even if the developers do not know the terms required to search for AK, because tags are linked to meta-tags that have a fixed meaning. Meta-tags could, therefore, be a guide to find AK, since they would not lose meaning overtime. However, we must conduct a long-term evaluation to better assert this. The evaluation results, therefore, show that the AK retrieval mechanism is (**RQ2**) suitable and feasible to help agile/global developers obtain AK from UTEM logs, because the mechanism is (1) useful and usable, and (2) the AK retrieval performance was better using the proposed mechanism when developers ignored which UTEM log contained the required knowledge.

Implementing the AK Condensation concept could help reduce AK vaporization in AGSD environments, taking into account the global and distribution aspects, without affecting the teams' agility. Furthermore, by reducing AK vaporization in AGSD, problems related to wasted time, software defects, and software projects' technical understanding [53,54] could be also reduced. It is worth recalling that we are presenting a single means to implement the concept of AK Condensation. Different implementations could be created solely by considering the basic items of this concept. It might be interesting to evaluate another implementation to confirm the feasibility of AK Condensation.

8. Conclusions and future work

In this paper, we present the concept of AK Condensation, which consists of structuring and retrieving AK shared by means of UTEM to reduce its vaporization in an AGSD environment. We also present an implementation of this concept, which was evaluated to determine AK Condensation feasibility. The evaluation results allowed us to determine that this concept could be feasible in AGSD environments.

On the one hand, these results could be attractive for AGSD practitioners, since an implementation of AK Condensation could reduce the amount of time wasted trying to find solutions to past problems, along with reducing the number of interruptions among teammates, since an additional source of AK would be available, i.e., an AK Condenser. In addition, AGSD practitioners might be interested in an implementation of AK Condensation because it could be a workaround to documentation debt, a means to alleviate architectural technical debt, and thus reduce AK vaporization. However, we are aware that there are cases of AGSD teams in which there is even a team of architects in charge of all the projects' architectural issues, as this is also reported in the empirical studies conducted by Clerc et al. [7], Razzak & Smite [66], and Alzoubi & Gill [67]. AK vaporization could, therefore, occur less frequently than in companies in which there is not a role or team that has these responsibilities. This leads us to believe that AK Condensation may be more appropriate for small and medium-sized AGSD companies,[7] which do not have sufficient resources and infrastructure to have an architect role.

[7] https://www.gartner.com/it-glossary/smbs-small-and-midsize-businesses.

On the other hand, our results may be interesting for software engineering researchers, since AK Condensation represents a convenient way in which to manage AK without much obstruction to the developers' work in AGSD. It may, therefore, be worth continuing exploring the UTEM logs as an AK source. Furthermore, AK Condensation represents a first step toward converting AK from tacit to explicit in a formalized manner [41] (e.g. UML notation), since AK would be structured by a classification scheme, which could ease the formal representation of this knowledge. Moreover, the impact of AK Condensation on the transitions between tacit and explicit knowledge and vice-versa (expressed using the SECI model [68]) could be explored in the future, owing to the close relation between AK Condensation and these two types of knowledge.

As future work, we shall improve ArchiKCo by using artificial intelligence to ease tag selection, adding a context aware suggestion feature to the Tagging Helper component. We shall also develop new versions of this component that will run with other UTEM, such as Trello, Slack, Jabber or Outlook, to be able to conduct studies in other contexts. AK Searcher must be adapted to these UTEM, since there will be synchronous and asynchronous media, and the frequencies of messages are different. Another improvement is the inclusion of a feature to exclude personal messages, such that only work-related messages would be considered during the search. In order to conduct evaluations in real scenarios, we must also include a strategy to motivate developers to tag. This strategy could be the following: when a developer finds a useful AK, s/he could qualify the message author, which could incentivize the best-qualified tagger. Finally, evaluations in real scenarios will allow us to refine the meta-tag model, and to observe how AK Condensation is conducted during pressure scenarios, thus enabling us to observe the implications as regards adopting the concept in AGSD.

Acknowledgments

The authors are grateful to the participating companies: EMCOR, Sahuaro Labs, Tufesa, Grupo SMI, Trapishar, Ubilogix and Softtek, for the support provided to carry out the present study, and for their willingness to continue working with us in future projects.

Funding

This work was supported by the National Council of Science and Technology (whose acronym in Spanish is Conacyt) of Mexico, with scholarship number 394125 for the first author. This work is also partially supported by: GINSENG (TIN2015-70259-C2-1-R, Ministerio de Economía y Competitividad y Fondo Europeo de Desarrollo Regional FEDER); and G3Softproject (SBPLY/17/180501/000150) funded by "Consejería de Educación, Cultura y Deportes de la Dirección General de Universidades, Investigación e Innovación de la JCCM" of Spain.

Conflict of interest

We have no conflict of interest to declare.

References

[1] B. Ramesh, L. Cao, K. Mohan, P. Xu, Can distributed software development be agile? Commun. ACM 49 (2006) 41, doi:10.1145/1164394.1164418.
[2] H. Holmstrom, E.O. Conchuir, P.J. Agerfalk, B. Fitzgerald, Global software development challenges: a case study on temporal, geographical and socio-cultural distance, Glob. Softw. Eng. (2006) 3–11 ICGSE '06. Int. Conf. (2006), doi:10.1109/ICGSE.2006.261210.
[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, The Agile Manifesto, (2001). http://agilemanifesto.org/.
[4] R. Hoda, J. Noble, S. Marshall, How much is just enough? in: Proc. 15th Eur. Conf. Pattern Lang. Programs - Eur. '10, New York, USA, ACM Press, 2010, p. 13, doi:10.1145/2328909.2328926.
[5] A. Cockburn, J. Highsmith, Agile Software Development:the People Factor, Computer 34 (2001) 131–133, doi:10.1109/2.963450.
[6] A.R. Yanzer Cabral, M.B. Ribeiro, R.P. Noll, Knowledge Management in Agile Software Projects: a Systematic Review, J. Inf. Knowl. Manag. 13 (2014) 1450010, doi:10.1142/S0219649214500105.
[7] V. Clerc, P. Lago, H. Van Vliet, Architectural knowledge management practices in agile global software development, in: 2011 IEEE Sixth Int. Conf. Glob. Softw. Eng. Work., IEEE, 2011, pp. 1–8, doi:10.1109/ICGSE-W.2011.17.
[8] M.A. Razzak, R. Ahmed, Knowledge sharing in distributed agile projects: techniques, strategies and challenges, in: 2014 Fed. Conf. Comput. Sci. Inf. Syst., Warsaw, Poland, IEEE, 2014, pp. 1431–1440, doi:10.15439/2014F280.
[9] E. Tom, A. Aurum, R. Vidgen, An exploration of technical debt, J. Syst. Softw. 86 (2013) 1498–1516, doi:10.1016/j.jss.2012.12.052.
[10] ISO/IEC/IEEE, Systems and software engineering – – Architecture architecture description, ISO/IEC/IEEE 420102011(E) (Revision ISO/IEC 420102007 IEEE Std 1471-2000). (2011) 1–46. doi:10.1109/IEEESTD.2011.6129467.
[11] P. Kruchten, P. Lago, H. van Vliet, in: Building Up and Reasoning About Architectgural Knowledge, 4214, 2006, pp. 95–110, doi:10.1007/11921998. Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics).
[12] A. Moraes, E. Silva, C. da Trindade, Y. Barbosa, S. Meira, Recommending experts using communication history, in: 2nd Int. Work. Recomm. Syst. Softw. Eng. - RSSE '10, 2010, pp. 41–45, doi:10.1145/1808920.1808929.
[13] M. Aniche, M.A. Gerosa, C. Treude, Developers' perceptions on object-oriented design and architectural roles, in: Proc. 30th Brazilian Symp. Softw. Eng., New York, NY, USA, ACM, 2016, pp. 63–72, doi:10.1145/2973839.2973846.
[14] I. Ghani, M. Bello, Agile adoption in IT organizations, KSII Trans. Internet Inf. Syst. 9 (2015) 3231–3248, doi:10.3837/tiis.2015.08.029.
[15] A.M.M. Hamed, H. Abushama, Popular agile approaches in software development: review and analysis, in: 2013 Int. Conf. Comput. Electr. Electron. Eng., 2013, pp. 160–166, doi:10.1109/ICCEEE.2013.6633925.
[16] H.M. Sneed, Dealing with Technical Debt in agile development projects, Lect. Notes Bus. Inf. Process. 166 (2014) 48–62 .LNBIP, doi:10.1007/978-3-319-03602-1.
[17] T. Clear, Documentation and Agile Methods: striking a Balance, SIGCSE Bull. 35 (2003) 12–13, doi:10.1145/782941.782949.
[18] H.-C. Estler, M. Nordio, C.A. Furia, B. Meyer, J. Schneider, Agile vs. structured distributed software development: a case study, in: Proc. IEEE Int. Conf. Glob. Softw. Eng., 2012, doi:10.1109/ICGSE.2012.22.
[19] G. Borrego, A.L. Morán, R. Palacio, O.M. Rodríguez, Understanding architectural knowledge sharing in AGSD teams: an empirical study, in: 2016 IEEE 11th Int. Conf. Glob. Softw. Eng., Los Alamitos, CA, USA, IEEE Computer Society, 2016, pp. 109–118. doi:doi.ieeecomputersociety.org/10.1109/ICGSE.2016.29.
[20] B. Selic, Agile documentation, anyone? IEEE Softw. 26 (2009) 11–12, doi:10.1146/annurev.ps.29.020178.002001.
[21] M.L.I. Gervigny, S.D. Nagowah, Knowledge sharing for agile distributed teams: a case study of Mauritius, in: 2017 Int. Conf. Infocom Technol. Unmanned Syst., Dubay , UAE, IEEE Computer Society, 2017, pp. 413–419, doi:10.1109/ICTUS.2017.8286043.
[22] J. Bosch, Software architecture: the next step, in: F. Oquendo, B. Warboys, R. Morrison (Eds.), EWSA, Springer, 2004, pp. 194–199.
[23] K. Dalkir, Knowledge Management in Theory and Practice, Second ed., The MIT Press, 2011.
[24] R. Farenhorst, R.C. de Boer, Knowledge management in software architecture: state of the art, in: M. Ali Babar, T. Dingsøyr, P. Lago, H. van Vliet (Eds.), Softw. Archit. Knowl. Manag. Theory Pract., Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 21–38, doi:10.1007/978-3-642-02374-3_2.
[25] S. Dorairaj, J. Noble, P. Malik, Knowledge management in distributed agile software development, in: 2012 Agil. Conf., Dallas, USA, IEEE, 2012, pp. 64–73, doi:10.1109/Agile.2012.17.
[26] M. Jiménez, M. Piattini, A. Vizcaíno, Challenges and improvements in distributed software development: a systematic review, Adv. Softw. Eng. 2009 (2009) 14, doi:10.1155/2009/710971.
[27] K.B. Awar, M.S.I. Sameem, Y. Hafeez, A model for applying Agile practices in Distributed environment: a case of local software industry, in: Proc. 2017 Int. Conf. Commun. Comput. Digit. Syst. C-CODE 2017, 2017, pp. 228–232, doi:10.1109/C–CODE.2017.7918933.
[28] T. Dingsøyr, Strategies and approaches for managing architectural knowledge, in: M.A. Babar, T. Dingsøyr, P. Lago, H. van Vliet (Eds.), Softw. Archit. Knowl. Manag., Springer, Berlin Heidelberg, 2009, pp. 59–68, doi:10.1109/ASWEC.2008.4483186.
[29] R. Capilla, A. Jansen, A. Tang, P. Avgeriou, M.A. Babar, 10 years of software architecture knowledge management: practice and future, J. Syst. Softw. 116 (2016) 191–205, doi:10.1016/j.jss.2015.08.054.
[30] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, M. Ali, A comparative study of architecture knowledge management tools, J. Syst. Softw. 83 (2010) 352–370, doi:10.1016/j.jss.2009.08.032.
[31] M. Galster, M.A. Babar, Empirical study of architectural knowledge management practices, in: 2014 IEEE/IFIP Conf. Softw. Archit., 2014, pp. 239–242, doi:10.1109/WICSA.2014.28.
[32] R. Farenhorst, R.C. De Boer, Knowledge management in software architecture: sState of the art, (n.d.) 21–39. doi:10.1007/978-3-642-02374-3.
[33] N. Ali, S. Beecham, I. Mistrik, Architectural knowledge management in global software development: a review, in: Glob. Softw. Eng. (ICGSE), 2010 5th IEEE Int. Conf., 2010, pp. 347–352, doi:10.1109/ICGSE.2010.48.
[34] S. Beecham, J. Noll, I. Richardson, N. Ali, Crafting a global teaming model for architectural knowledge, in: Proc. – 5th Int. Conf. Glob. Softw. Eng. ICGSE 2010, 2010, pp. 55–63, doi:10.1109/ICGSE.2010.15.

[35] P. Gaubatz, I. Lytra, U. Zdun, Automatic enforcement of constraints in real-time collaborative architectural decision making, J. Syst. Softw. 103 (2015) 128–149, doi:10.1016/j.jss.2015.01.056.
[36] S. Sherman, I. Hadar, M. Levy, N. Unkelos-Shpigel, Enhancing software architecture via a knowledge management and collaboration tool, in: S. Kunifuji, G.A. Papadopoulos, A.M.J. Skulimowski, K. Janusz (Eds.), Knowledge, Inf. Creat. Support Syst. Sel. Pap. from KICSS'2014 – 9th Int. Conf. Held Limassol, Cyprus, Novemb. 6-8, 2014, Cham, Springer International Publishing, 2016, pp. 537–545, doi:10.1007/978-3-319-27478-2_41.
[37] M. Che, D.E. Perry, G. Yang, Evaluating architectural design decision paradigms in global software development, Int. J. Softw. Eng. Knowl. Eng. 25 (2015) 1677–1692, doi:10.1142/S0218194015400380.
[38] C. Yang, P. Liang, P. Avgeriou, A systematic mapping study on the combination of software architecture and agile development, J. Syst. Softw. 111 (2016) 157–184, doi:10.1016/j.jss.2015.09.028.
[39] M.A. Babar, A.W. Brown, I. Mistrik, Agile Software Architecture: Aligning Agile Processes and Software Architectures, Elsevier Inc., 2013, doi:10.1016/C2012-0-01208-2.
[40] G. Borrego, A.L. Morán, R. Palacio, O.M. Rodríguez-Elias, E. García-Canseco, Review of approaches to manage architectural knowledge in Agile Global Software Development, IET Softw. 11 (2017) 77–88, doi:10.1049/iet-sen.2016.0197.
[41] I. Nonaka, H. Takeuchi, The Knowledge-Creating company: How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, New York, 1995.
[42] H. Richter, G. Abowd, C. Miller, H. Funk, Tagging knowledge acquisition sessions to facilitate knowledge traceability, Int. J. Softw. Eng. Knowl. Eng. 14 (2004) 3–19, doi:10.1142/S0218194004001543.
[43] E. Bagheri, F. Ensan, Semantic tagging and linking of software engineering social content, Autom. Softw. Eng. 23 (2016) 147–190, doi:10.1007/s10515-014-0146-2.
[44] C. Treude, M.A. Storey, How tagging helps bridge the gap between social and technical aspects in software development, in: Proc. – Int. Conf. Softw. Eng., 2009, pp. 12–22, doi:10.1109/ICSE.2009.5070504.
[45] M.A. Storey, J. Ryall, J. Singer, D. Myers, L.T. Cheng, M. Muller, How software developers use tagging to support reminding and refinding, IEEE Trans. Softw. Eng. 35 (2009) 470–483, doi:10.1109/TSE.2009.15.
[46] S.M. Sohan, M.M. Richter, F. Maurer, Auto-tagging emails with user stories using project context, in: Lect. Notes Bus. Inf. Process. 48 LNBIP, 2010, pp. 103–116, doi:10.1007/978-3-642-13054-0_8.
[47] J.M. Al-kofahi, A. Tamrawi, T.T. Nguyen, H.A. Nguyen, T.N. Nguyen, Fuzzy set approach for automatic tagging in evolving software, Softw. Maint. (ICSM), 2010 IEEE Int. Conf., IEEE, 2010, doi:10.1109/ICSM.2010.5609751.
[48] S. Jalali, C. Wohlin, Global software engineering and agile practices: a systematic review, J. Softw. Evol. Process. 24 (2012) 643–659, doi:10.1002/smr.561.
[49] M. Levy, O. Hazzan, Knowledge management in practice: the case of agile software development, in: 2009 ICSE Work. Coop. Hum. Asp. Softw. Eng., 2009, pp. 60–65, doi:10.1109/CHASE.2009.5071412.
[50] J. Shore, S. Warden, The Art of Agile Development, First, O'Reilly, 2007.
[51] E. Isaacs, S. Whittaker, D. Frohlich, ... B. O'Conaill, Informal communication re-examined: new functions for video in supporting opportunistic encounters, Mediat. Commun. (1997) 1–30.
[52] C. Gutwin, S. Greenberg, M. Roseman, Workspace awareness in real-time distributed groupware: framework, widgets, and evaluation, in: M.A. Sasse, R.J. Cunningham, R.L. Winder (Eds.), People Comput. XI, Springer London, London, 1996, pp. 281–298, doi:10.1007/978-1-4471-3588-3_18.
[53] H. Holz, G. Melnik, M. Schaaf, Knowledge management for distributed agile processes: models, techniques, and infrastructure, in: Enabling Technol. Infrastruct. Collab. Enterp. 2003. WET ICE 2003, IEEE, 2003, pp. 291–294, doi:10.1109/ENABL.2003.1231423.
[54] N. Uikey, U. Suman, ... A. Ramani, A documented approach in agile software development, Int. J. Softw. 2 (2011) 13–22.
[55] Z. Li, P. Liang, P. Avgeriou, Architectural debt management in value-oriented architecting, Econ. Softw. Archit. (2014) 183–204, doi:10.1016/B978-0-12-410464-8.00009-X.
[56] S. Ryan, R.V. O'Connor, Acquiring and sharing tacit knowledge in software development teams: an empirical study, Inf. Softw. Technol. 55 (2013) 1614–1624, doi:10.1016/j.infsof.2013.02.013.
[57] M.A. Babar, Supporting the software architecture process with knowledge management, Softw. Archit. Knowl. Manag. Theory Pract. (2009) 69–86, doi:10.1007/978-3-642-02374-3_5.
[58] T. Zernadji, C. Tibermacine, F. Cherif, Processing the evolution of quality requirements of web service orchestrations: a pattern-based approach, in: Proc. – Work. IEEE/IFIP Conf. Softw. Archit, 2014, pp. 139–142, doi:10.1109/WICSA.2014.35. WICSA 2014. (2014).
[59] G. Borrego, A.L. Morán, R. Palacio, Preliminary evaluation of a tag-based knowledge condensation tool in agile and distributed teams, in: 2017 IEEE 12th Int. Conf. Glob. Softw. Eng., 2017, pp. 51–55, doi:10.1109/ICGSE.2017.14.
[60] C. Wohlin, P. Runeson, M. Hst, M.C. Ohlsson, B. Regnell, A. Wessln, Experimentation in Software Engineering, Springer Publishing Company, Incorporated, 2012.
[61] J. Brooke, SUS-A quick and dirty usability scale, Usability Eval. Ind. 189 (1996) 194, doi:10.1002/hbm.20701.
[62] F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Q. 13 (1989) 319–340, doi:10.2307/249008.
[63] J. Sauro, J.R. Lewis, Quantifying the User Experience: Practical Statistics for User Research, first ed., Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2012.
[64] A. Bangor, P.T. Kortum, J.T. Miller, An empirical evaluation of the system usability scale, Int. J. Hum. Comput. Interact. 24 (2008) 574–594, doi:10.1080/10447310802205776.
[65] U. Dekel, J.D. Herbsleb, Pushing relevant artifact annotations in collaborative software development, in: Proc. ACM 2008 Conf. Comput. Support. Coop. Work – CSCW '08, 2008, p. 1, doi:10.1145/1460563.1460565.
[66] M.A. Razzak, D. Smite, Knowledge management in globally distributed agile projects – lesson learned, in: Glob. Softw. Eng. (ICGSE), 2015 IEEE 10th Int. Conf., Ciudad del Real, Spain, 2015, pp. 81–89, doi:10.1109/ICGSE.2015.22.
[67] X. Shen, Y. Li, Y. Sun, An agile enterprise architecture-driven model for geographically distributed agile development, transform, Healthc. Through Inf. Syst. 17 (2016) 185–197, doi:10.1007/978-3-319-30133-4.
[68] I. Nonaka, R. Toyama, N. Konno, SECI, Ba and leadership: a unified model of dynamic knowledge creation, Long Range Plan. 33 (2000) 5–34, doi:10.1016/S0024-6301(99)00115-6.
[69] F. Calefato, D. Gendarmi, F. Lanubile, Investigating the use of tags in collaborative development environments: a replicated study, in: Proc. 2010 ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas, 24, 2010, pp. 1–24, doi:10.1145/1852786.1852818. 9.
[70] A. Forward, T. Lethbridge, D. Deugo, CodeSnippets plug-in to eclipse: introducing web 2.0 tagging to improve software developer recall, in: Proc. – SERA 2007 Fifth ACIS Int. Conf. Softw. Eng. Res. Manag. Appl., 2007, pp. 498–502, doi:10.1109/SERA.2007.81.
[71] S. Paul, T. Makkar, K. Chandrasekaran, Software development using context aware searching of components in large repositories, in: Int. Conf. Comput. Commun. Autom., IEEE, 2015, pp. 765–772, doi:10.1109/CCAA.2015.7148513.