IET Software

Special Issue: Knowledge Discovery for Software Development (KDSD)

Ontology for knowledge condensation to support expertise location in the code phase during software development process

ISSN 1751-8806
Received on 1st September 2019
Revised 17th December 2019
Accepted on 27th January 2020
E-First on 8th April 2020
doi: 10.1049/iet-sen.2019.0272
www.ietdl.org

Jose R. Martínez-García[1], Francisco-Edgar Castillo-Barrera[1], Ramon R. Palacio[2] ✉, Gilberto Borrego[3], Juan C. Cuevas-Tello[1]

[1]*Computer Science, Autonomous University of San Luis Potosi, San Luis Potosí, Mexico*
[2]*Unidad Navojoa, Instituto Tecnológico de Sonora, Navojoa, Mexico*
[3]*Computación y Diseño, Instituto Tecnológico de Sonora, Cd. Obregón, Mexico*
✉ *E-mail: ramon.palacio@itson.edu.mx*

Abstract: Software Development is a complex process, in which every software product is a knowledge representation of all the involved people. In agile software development, knowledge is prone to vaporise, because documentation is not a priority as indicated in the agile manifesto. This condition generates problems such as poor understanding of the requirements, knowledge transfer deficiency among developers, time wasted by developers while searching for knowledge. The objective of this work is to reduce *architectural knowledge vaporisation* by means of knowledge condensation to support *expertise* location (high-level knowledge at a given time). This through an ontology that will condensate the knowledge in the code phase. This study presents the description of an ontology development process following the Methontology Framework. Results show that the proposed ontology does not present incongruence or inconsistency and answers the competency questions correctly. The main contribution of this study is the ontology which brings several benefits such as a shared concept of the knowledge in the code phase and a way to link the artefacts (resources used by developers in the project) and the *experts* (artefacts provider).

1 Introduction

Software Development is considered a complex process, in which every software product is a representation of the knowledge of all the involved people, due to this knowledge is used to solve client's needs by generating a computer application [1, 2]. For this reason, development teams require constant interaction with the project's stakeholders, since the teams are integrated by people working in different phases and activities. Particularly, coding is a phase of the software development cycle, aimed to translate the system design into code in a given programming language. During this phase, a lot of knowledge can be shared among the team members [3], for example, programmers and testers.

In software companies, knowledge can be found in two sources [4]: (i) *artefacts* such as documentation [5] (e.g. requirements document, vision document), source code [6], repositories where developers store and consult digital (e.g. blogs [7, 8], bookmarks) or physical documents (e.g. manuals) and project management tools; (ii) *persons* with a certain level of knowledge in a specific area that could be useful to solve problems during development tasks [9, 10]. Therefore, anyone in a company could be a potential source of knowledge.

In agile software development, documentation is not a priority, as expressed in agile manifesto [11], thus, it is reflected in agile software projects with minimal documentation. This situation is caused because the face to face interaction is preferred by developers to clarify doubts or to solve problems. Hence, knowledge is prone to vaporise due to the manifest's principles [12, 13]. Borrego *et al.* [13] define *architectural knowledge vaporisation* as the loss of artefacts and architectural documents owed to poor documentation. The fact of not having this knowledge generates the following problems [14, 15]: (i) poor understanding of the requirements and technical solutions; (ii) knowledge transfer deficiency among developers; (iii) evolution and maintenance drawbacks; and (iv) time wasted by developers while searching for artefacts or experts. These problems cause the increase of time and cost in software development projects [16].

To address these problems, developers seek high-level knowledge at a given time (*expertise*) [10], through which developers could resolve problems or doubts that arise daily during their workday to help them fulfil their activities [17]. However, developer teams generally do not benefit from this knowledge (*artefacts* and *experts*), generated during the software development process. Artefacts usually are not linked with their creators, moreover each person labels or stores knowledge in different ways. In this case, artefacts become hard to identify, because they are not available or they get lost (when the provider/creator leaves the company) [18–21].

Therefore, it is necessary to link the artefacts generated during the coding phase with the creator (*experts*). The goal is that artefacts can be easier to find (*expertise*) and accessible to everyone, even when the knowledge provider or the expert is not present in the company. Thus, knowledge needs to be condensate. Borrego *et al.* [13] define the *knowledge condensation* as the process of capturing and classifying knowledge before it loses, where the objective is to ease knowledge retrieval.

This is an important issue because to make good decisions when having a doubt or problem developers need reliable and precise information (*expertise*) [22], developers have an expertise need and this can be found in different sources as mentioned before.

In our approach, the knowledge condensation is done by using ontologies, which are a formal description of a shared concept [23], and they allow to define a vocabulary to share information in a certain domain. Moreover, in addition, to help with the decision making, ontologies could mitigate the problems caused by the knowledge vaporisation mentioned before, and also could prevent constant questions to experts which sometimes lead to an erosion in interpersonal relationships, affecting the knowledge flow. Additionally, ontologies present a visual way to share a common understanding of an information structure between several people or computer systems [24], and also they allow to reuse knowledge through ambiguities clarification [25].

As we can notice, an ontology can provide benefits in software development through an information structure that can do automated reasoning about knowledge in the software development life cycle (e.g. project management [26], software measurement

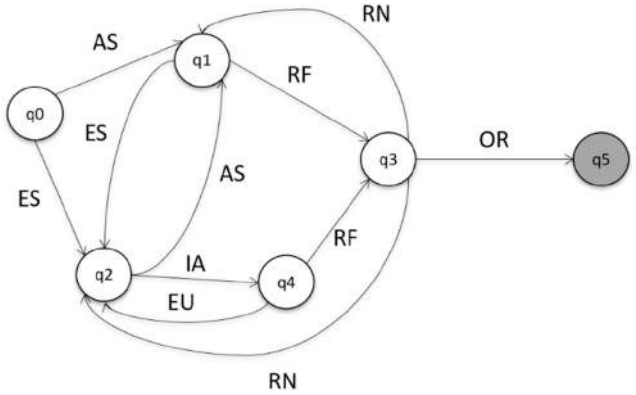


Fig. 1 *Initial state is q_0, which represent the need of expertise to solve a doubt or problem, and q_5 is the final state, representing when a doubt or problem was solved*

Table 1 Ontologies supporting software development activities

Stage/ Ontology	Specification	Conceptualisation	Formalisation
requirements	documentation of experiments in distributed software [46]	documentation of term and concepts in GSD [47, 48]	human resources assign [26]
	sutomatic software documentation [49]	effort Estimation [50]	effort estimation [51]
	software requirements [42, 43]	documentation of terms and concepts in multisite development [52, 53]	software requirements [41, 44, 45]
		documentation for task allocation [54]	documentation to support distributed teams [5]
		software measurement [27]	
		service management [55]	
design		modelling software process [56]	
coding			
testing			
maintenance	knowledge management [28]		software artefacts traceability [57]

[27], maintenance [28]). In this sense, the challenge is getting a described content capable of: (i) distinguish among different word meanings; (ii) infer relations between words on a specific thematic context; and (iii) retrieve information according to the user needs. Thus, information must be described and classified in a way that it can be understandable for a computer or any user within an organisation.

In Fig. 1, we present automata to illustrate a scenario of the *expertise* location. The transitions are objective reached (OR), artefact seeking (AS), expert seeking (ES), identifying availability (IA), resource need (RN), expert unavailable (EU), a resource founded (RF). The automata have six states $\{q_0, \ldots, q_5\}$. This scenario starts with a developer having the need of expertise to solve a problem or doubt (q_0). Then, the developer can choose between an expert search (q_2) or an artefact search (q_1). The developer searches for all available artefacts, in the case of choosing an artefact search (q_1). An artefact verification is done to see if it is fulfilled the objective (q_3). If the need for expertise is satisfied, the problem is solved (q_5). On the contrary, if the need for expertise is not satisfied the developer must check more artefacts to solve the problem (q_1). On the other hand, in the case of choosing an expert search (q_2), the developer does a search looking for someone with the knowledge to solve the problem or doubt, once the right person is found, his/her availability must be checked (q_4).

Finally, if the expert fulfilled the objective, the problem is solved (q_5). Otherwise, another expert search is needed (q_2). Sometimes an expert can lead the requester to an artefact (q_1) and vice-versa (q_2).

As we can see, *expertise* location is a complex process, since despite that *expertise* exists in an organisation, this is not always accessible. Sometimes an expert is unavailable for diverse reasons, skips the day, is on vacation, leaves the company etc. Therefore, artefacts used or created by an expert may not be accessible.

From the above, the main objective of this work is to describe the development process and validation of a domain ontology, which is focused on the coding phase of a software development process. The aim is to link artefacts and experts through an information structure (ontology).

The rest of the paper is structured as follows: Section 2 shows the related works of software engineer ontologies; Section 3 presents the methodology followed to develop the ontology; in Section 4 we present the results obtained following the methodology; Section 5 presents the validity threats of this work; finally, in Section 6 we present the discussion and conclusion of our results and we present our acknowledgments in Section 7.

2 Related work

Recently, Software Engineers have had an interest in the use of ontologies to identify and share knowledge on the different phases of the software development cycle as presented in the works by Bathia *et al.* [29] and de Souza *et al*. [30]. In the literature, we can find different methodologies to develop ontologies including Diligent [31], On-To-Knowledge [32], Neon [33], OntoDocMan [34] and Methontology Framework [35], where each one follows a different approach.

The Methontology Framework is widely used because its resemblances the software development process in software engineering. Since the approach is focused on software engineering, it will easy to follow by software developers. Some authors refer to the Methontology Framework, e.g. an ontology was developed to represent college graduation screening process [36], the development of STIFIn Ontology Finger Personality Solution [37], among others.

The Methontology Framework consists of the following phases: (i) *Specification:* identification of the domain, goal, relevant terms, and objectives; (ii) *Conceptualisation:* creates a glossary of terms and from this create a taxonomy; and (iii) *Formalisation:* creates a formal computational model using a tool (e.g. Protégé [38]).

Previous to the ontology development, we conducted a literature review of ontologies in the software development process. The ontologies found during the literature review have addressed some problems related to some specific development phase. These are: Requirements, Design, Implementation, Testing and Maintenance. These phases are the same regardless of the methodology used and are known as the software development life cycle [39, 40].

Table 1 shows the ontologies found as a result of a literature review. These ontologies were classified based on the software development cycle phase in which they are focus and how much progress they achieve based on the Methontology Framework. Almost all the ontologies work on the requirements phase, targeting activities such as requirement analysis. In [41–45] proposals seek to enhance the software quality by improving the elicitation and administration of the user's needs. Their objective was to eliminate ambiguities and language mistakes. Additionally, they elicit the requirements in a faster way and obtain an output in different formats (e.g. UML, data models, and diagrams).

Another activity performed in the requirements phase is *effort estimation*. Hamdan *et al*. [50] seek to eliminate mistakes or

misunderstandings through the use of ontologies in a system, which enables project managers to obtain characteristics and terms of previous projects. Adnan and Afzal [51] present an ontology for knowledge about effort estimation, which has the objective of mitigate problems of distributed teams (e.g. inaccurate estimation of effort and time) by making an estimation based on previous projects. Pre-established terms are used to describe the project.

Documentation is another activity addressed in the requirements phase. Rocha and Meira [46] present a literature review of ontologies focused on documentation in distributed software development (DSD). Their objective is to propose a recommendation system, based on the issues founded in the literature review. Addressing the same activity, Bathia *et al.* [49] present a conceptual description of an ontology, to get automatic documentation through a system that uses an ontology. This ontology establishes terms and concepts to be used across projects. There are ontologies which work with documentation activities [47, 48, 52, 53, 58], which try to create a semantic (generalised interpretation) between concepts and terms used either in Global Software Development (GSD) or DSD. Their objective is to describe GSD o DSD projects.

Finally, Marques *et al.* [54] present an ontology to document task allocation in distributed software development teams. This ontology offers a reference to concepts and terms used in distributed software development, and also tasks and information of the developers (e.g. projects, country, time zone, experience).

Another activity included in the requirements phase is *human resources assign*. Paredes-Valverde *et al.* [26] present an ontology that broadly describes user data (developers). The aim is to assign developers into projects according to their experience and the needs of the project. On the other hand, Fonseca *et al.* [27] present an ontology for software measurement activity (e.g. measure source code, effort estimation, number of sprints). This ontology aims to unify the outputs of different tools that measure software. The goal was to have a better measurement using the data from all the measurement tools. Finally, Valiente *et al.* [55] present an integration model between Software Engineering and technology administration service. This model uses an ontology to integrate terms and concepts involved within these areas. The purpose was to make clear to the client his options, and the client's needs to the developer.

Regarding the design phase, Martinho *et al.* [56] present an ontology for modelling software processes. The ontology was designed using knowledge modelling tools (Cmpas & CmapsTools). The ontology is focused on the project flexibility design (information that the developers want to use based on their experience).

Finally, in the maintenance phase, Serna and Serna [28] present a conceptual analysis of applying ontologies for knowledge management in the maintenance phase during software development. Instead of conceptual analysis, Zhang *et al.* [57] present a maintenance ontology for tracking artefacts to link them with the software requirements, with the main purpose of differentiate a code mistake from a requirement mistake.

Undoubtedly, there are many software development activities that can be benefited with the support of ontologies, but most of these have not been formalised, in consequence, these ontologies are not properly appreciated. In Table 1, it can be noticed that there exists an area of opportunity to build ontologies in the testing and coding phases. Particularly, an ontology in coding phase could bring several benefits, such as support to the expertise location by generating concepts and terms of the knowledge in this phase with the purpose of linking the artefacts with their provider. Furthermore, the ontology could generate knowledge representation, this representation will help to identify resources used in a project and their providers. In this way, the information (*expertise*) will be more accessible, which will reduce the time invested in the expertise location process (see Fig. 1). In addition to this, most of the work presented in this section is not focused on the code phase. In some cases, these ontologies are presented as a conceptual model.

The next section presents the formal methodology that was used to define our ontology.

3 Method

This research followed the Methontology Framework [35] which is an accepted methodology to define the development life cycle in Ontological Engineering (from requirements specification to maintenance). The Methontology Framework life cycle (Fig. 2) includes five phases: (i) Specification; (ii) Conceptualisation; (iii) Formalisation; (iv) Evaluation; and (v) Maintenance. The next subsections describe the phases and activities needed for the development of our ontology.

3.1 Specification phase

Specification phase establishes a document covering the ontology's purpose (i), scope (ii), implementation language (iii), intended End-Users (iv) and Intended Uses (v). This document is done by doing the following task.

3.1.1 Knowledge acquisition activity: In software development, the key to project success lies in the software specification [39]. Suarez-Figueroa *et al.* [59] present guidelines based on the use of the Competency Questions (CQ) and the existing methodologies to build ontologies. These guidelines help to capture knowledge from users and to produce the Ontology Requirement Specification Document (ORSD). The ORSD document helps to identify the knowledge that the ontology contains, and it is useful to define the requirements the ontology must cover.

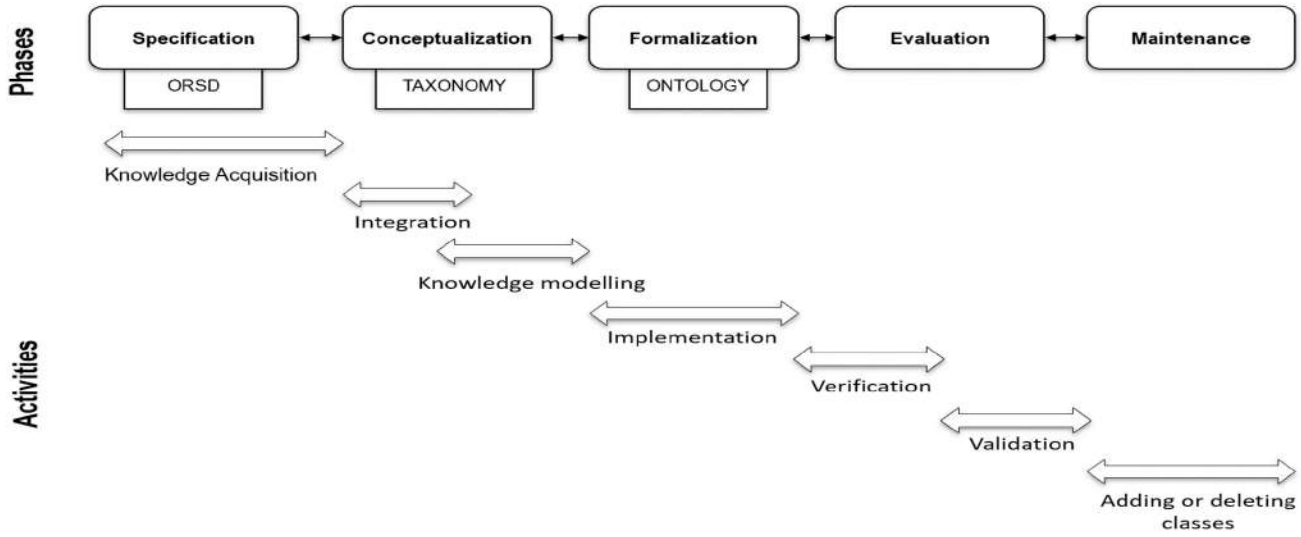


Fig. 2 *Methontology Framework life cycle*

For the knowledge acquisition activity, we come up with three approaches: (i) interviews; (ii) focus group; (iii) survey form. Three different groups were selected to cover different approaches. The participants were developers from different organisations (e.g. software organisations and organisations with a software developer department).

The first approach carried out was the interviews, the objective was to know the way developer teams work, (e.g. communication, challenges or problems). The interviews contained questions related to general information (role, activities and organisation), project administration, version control, coordination and team mindset.

The objective of the focus group was to know the process of searching for expertise within the software development teams either to store it or to share it, as well as the process of finding an expert for consulting. The participants were asked about the process of individual search, knowledge sharing and expert search.

Finally, the last approach, survey form, was applied to group developers. The objective was to obtain examples of problem or doubts that the developers usually try to solve. The survey contains the following fields: search keywords, type of search (expert or artefacts) and time invested trying to solve a doubt or problem.

3.2 Conceptualisation phase

Once all the needed knowledge has been acquired, it must be organised. Conceptualisation phase is focused on organising and structuring the acquired knowledge using external representations (e.g. UML, IDEF5) which are independent of the ontology implementation languages. The organising and structuring tasks are as follows.

3.2.1 Integration activity: To avoid redundant information, it must be considered the reuse of ontologies (definitions already built). The Ontologies were consulted in the following databases:

- Swoogle [60].
- DAML Ontology Library [61].
- ONKI Ontology Library Service [62].
- Linked Open Vocabularies [63].

These databases were searched on the internet and some of them are the most commonly cited in research articles available. In addition, we searched for ontologies of the same domain in academic databases (e.g. IEEE Xplore, ScienceDirect), like the one trying to build in this work in academic databases.

3.2.2 Knowledge modelling activity: This task consists of storing statements about facts by building meaningful information structures through multiple representations (e.g. mind maps).

3.3 Formalisation phase

Formalisation phase converts a conceptual model (taxonomy) to a formal model (computable). Specifically, for this work we took the taxonomy created in the conceptualisation phase, it was converted using to a computation model using Protégé tool [38]. This activity is known as Implementation activity.

3.4 Evaluation phase

In the traditional Methontology Framework the evaluation is considered as an activity which is carried out during all the phases. In our work, this activity is considered as another phase in the proposed methodology, which consists of carrying out a technical judgement of the ontology, according to the ORSD, by doing the following tasks.

3.4.1 Verification activity: This activity is a technical process which is done to guarantee the correctness of the ontology, according to the specification requirements. The verification activity was being done using the Pellet plugin reasoner on Protégé. An ontology reasoner is a piece of software able to infer logical consequences from a set of asserted facts or axioms.

3.4.2 Validation activity: It is the process done to ensure that the ontology fulfils the purpose for which it was built. The validation was being done by using CQ [64], which consists in a set of questions defined in the ORSD in a natural language, the ontology must answer these questions correctly. The CQs were based on the survey form that the developers fill with examples of doubts of problems that they try to solve.

Finally, at the end of all the phases, there is the maintenance phase, which consists of tasks covering from erasing obsolete instances or adding new ones over time. This phase was considered because the scope of we were focused only on ontology development.

4 Results

Here, we present our results obtained by following the phases and activities defined above.

4.1 Specification phase results

The knowledge acquired was collected following three approaches: interview, focus group and survey form (see Section 3.1). In the interview participated 6 developers and 2 project managers from 6 different software development companies. In the focus group participated 4 developers and 1 designer from the same company. Finally, in the survey form participated 12 developers from 3 different companies.

The data were extracted from the interviews, focus group and survey from using affinity diagrams, which is a tool that synthesises a set of verbal data (e.g. ideas, opinions, expressions) grouping them according to the relationship they have with each other. This process begins with the transcription of the interviews to find the key data of the participants' responses. From that, the data of the answers that appeared most recurrently were classified. Later we continue with the analysis of the data to identify the relationships between the processes of the search for expertise. Finally, from the affinity diagram and the defined categories, conclusions were obtained. With the collected information as part of the specification phase, we create a document following the ORSD guidelines. Table 2 shows a fragment of the developed ORSD.

4.2 Conceptualisation phase results

As a result of the integration task, it was not found any ontology with the same domain as the one trying to build in this work, neither in ontology databases nor in the literature review.

Table 2 ORSD Fragment

Ontology Requirements Specification Document Template
1 Purpose
the integration of the artefacts, projects and experts in the code phase of the software development process
2 Scope
the ontology has a focus just on the code phase of the software development process domain. The level of granularity is directly related to the competency questions and terms defined
3 Implementation Language
the ontology must be implemented in OWL language using protégé ontology tool.
4 Intended End-Users
user 1. Programmer searching for resources to solve a problem (e.g. requirements, bugs, or doubts with a process)
user 2. Programmer searching for an expert to ask for help
user 3. Programmer searching for information about a project and his participants
user 4. Programmer updating or registering his expertise (projects or resources)

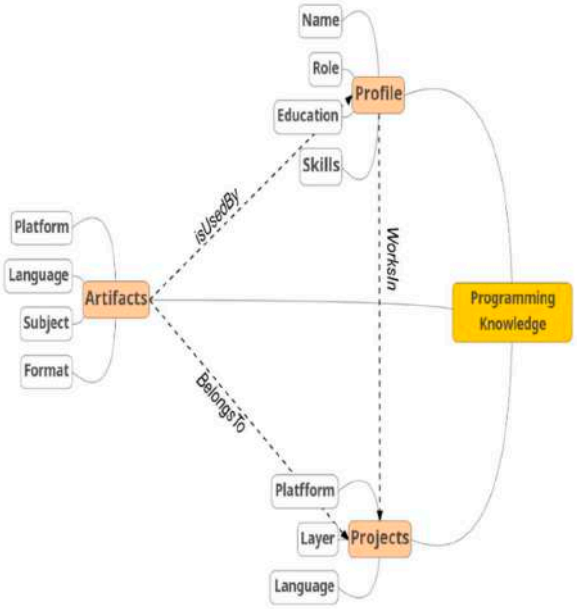


Fig. 3 *Taxonomy of Knowledge Expertise in code phase*

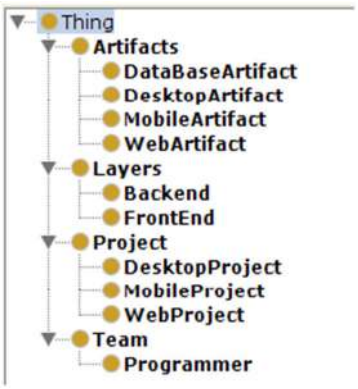


Fig. 4 *Coding phase expertise ontology: screenshot of main classes developed in Protégé*

Table 3 Ontology properties, ranges and domains

Property	Inverse	Type	Domain	Range
isUsedBy	hasUsed	functional	artefacts	programmer
isMemberOf	N/A	functional	programmer	projects
isMadeBy	hasMadeBy	functional	artefacts	programmer
hasWorked	N/A	functional	programmer	projects
hasUsedIn	N/A	functional	artefacts	projects
isBasedOn	N/A	functional	projects	layers

```
INFO  14:02:43 ------------------------------ Running Reasoner -------
INFO  14:02:43 Pre-computing inferences:
INFO  14:02:43     - class hierarchy
INFO  14:02:43     - object property hierarchy
INFO  14:02:43     - data property hierarchy
INFO  14:02:43     - class assertions
INFO  14:02:43     - object property assertions
INFO  14:02:43     - same individuals
INFO  14:02:43 Ontologies processed in 236 ms by Pellet
INFO  14:02:43
```

Fig. 5 *Pellet reasoner output*

Based on the terms and concepts identified from interviews, focus and a survey form, we created a taxonomy (see Fig. 3).

Fig. 3 shows a taxonomy of the knowledge produced in the code phase of the software development (programming knowledge). The main elements of the taxonomy are (i) profile; (ii) projects; and (iii) artefacts.

The Profile entity represents a description of a programmer in an organisation with information such as name, role, skills, projects has worked or is working currently. The Project entity represents information about the developer's current project. In this way, you can know the developers' skills based on the project history and the artefacts used in those projects. So, developers create artefacts by working on projects, and those are used by others to solve problems.

In summary, programmers have a profile and work assigned in a project, which is developed in a certain platform (e.g. web, database, desktop and mobile) which has layers (e.g. backend and frontend) and a programming language (e.g. JavaScript).

4.3 Formalisation phase results

Protégé tool [38] was used to convert the conceptual model to a computable model. This tool uses Ontology Web Language (OWL) [65] to define an Ontology.

Using the taxonomy (see Fig. 3), we defined the classes' names (in OWL, classes are interpreted as a set of individuals or objects), properties, and instances. The principal class that represents a set of all individuals is 'Thing', thus all classes are subclasses of that one (see Fig. 4). Fig. 4 shows the main classes of our ontology: Team, Artefacts, Project, Layers. Team class represents a developer team in an organisation. Artefact class represents the resources used by developers to solve a doubt or a problem. Project class represents a description of the work and activities done by developers.

In conclusion, developers (members of a Team class) work in a Project in an organisation, and when a developer has a doubt or problem uses Artefacts.

Properties in OWL represent a relationship between two individuals. There are two types of properties: object and data type properties. The object properties link an individual to another individual. The datatype properties link an individual to a data value expressed in Extensible Markup Language (XML) or Resource Description Framework.

The properties defined to our ontology are shown in Table 3. The property 'isUsedBy' help to link a programmer (a subclass of a team class) to the artefacts that have been used to solve problems or doubts in a project. The 'isMemberOf' and 'hasWorked' property helps to identify in which Programmer has worked or which project is currently working on. The properties 'IsMadeBy' and 'hasUsedIn' help to identify who creates an artefact and in which project was created or used.

4.4 Evaluation phase results

In the aim to perform the evaluation, the ontology must be populated by creating instances. This process usually involves linking data to the elements of the ontology. The instances were created from the participants' data.

As part of the verification activity, we used the Pellet reasoner. No incongruence or inconsistency was found in the ontology, when it was analysed with the reasoner (see Fig. 5).

In the case of the validation activity, the CQ (see Table 4) were used. Before that, questions must be transformed into a computer language, using the Manchester OWL syntax [66] to translate the questions in natural language into a computer language applied in Protégé.

Table 4 shows the questions designed to query in the validation activity. The questions are divided into two groups: (a) Expert seeking; (b) Artefact seeking. Due to the two types or searches done to solve a doubt or problem (see Fig. 1). You can either look for a resources (artefacts) or look for an expert the recommends you an artefact.

Fig. 6 presents a description of a scenario application used in the evaluation phase.

Fig. 7 shows an example of instances created during the ontology population. These instances represent a scenario of a

programmer working in an organisation. Omar represents an instance from the *Programmer* subclass, Project_One an instance from the *Project* class, and all the resources are instances from the *Artefacts* class. *Omar* is currently working on *Project_One* and has used many resources (*artefacts*) to solve doubts or problems in the project.

Table 4 Competency Questions in natural language

Competency Questions (CQs)
CQG1(Expert seeking)
CQ1. In which projects 'developer name' has been worked?
CQ2. In which language 'developer name' programs?
CQ3. Developers with skills on Java?
CQ4. Which resources has been used by 'developer name'?
CQG2(Artefact seeking)
CQ1. Resources for web developing?
CQ2. Resources used in 'name' project?
CQ3. Resources used by 'developer name' in 'name' project?

> *"Ana is a developer within an organization, she is a new member of the Project_one, one day she had a problem with one of the modules she was programming, so she had to spend time searching the web to solve the problem. Sometime later he met Omar another member of the Project_One, he told her that he had had the same problem, which would have been useful since he had the resources of how he had solved the problem."*

Fig. 6 *Scenario description*

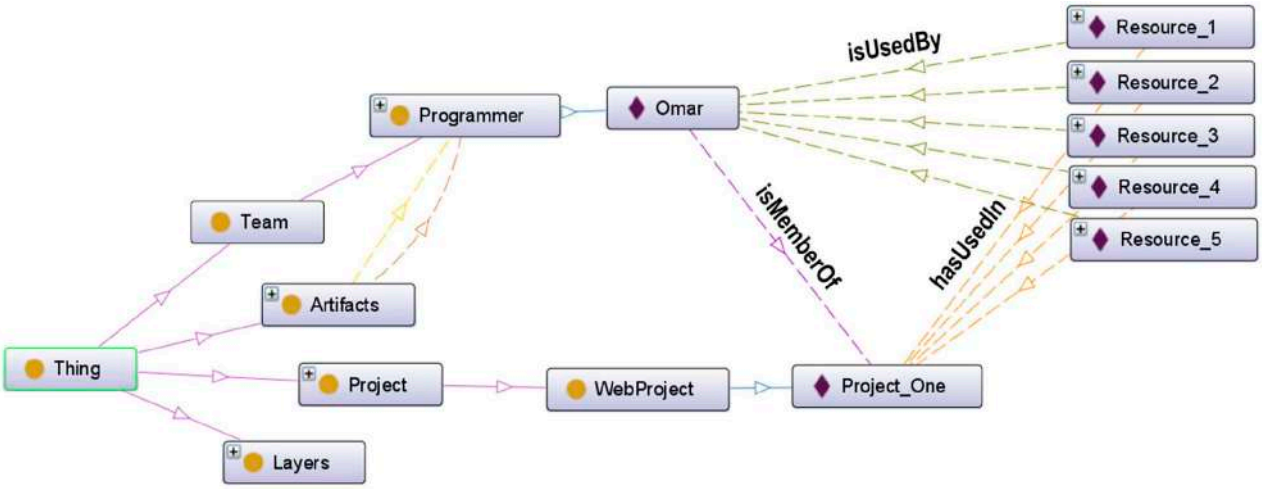


Fig. 7 *Ontology instances example*

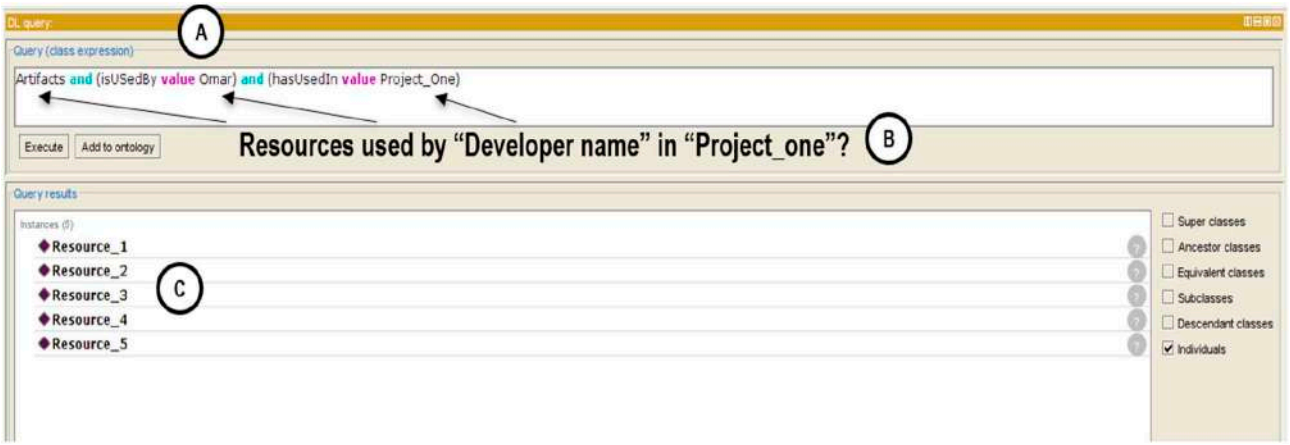


Fig. 8 *Competency Question example with Manchester OWL Syntax in Protégé*

Fig. 8 presents an example of a question done in Protégé during the validation activity. In this case, the object properties link the resources that *Omar* used in the *Project_One*. In this way, *Ana* could reuse the resources used by *Omar*, since Omar's resources will be associated with him and the project in which he used them.

5 Threats to validity

This study considers threats to the internal, external and conclusion validities [67].

Internal validity refers to the capacity to repeat the same behaviour on a new experiment considering the same participants. Communication and information sharing between participants is the main threat, which was mitigated by sending a survey form to the participants, to obtain examples or doubts and problems that the developers usually try to solve. Thus, the participants could fulfil anytime during the day. In addition, none of the participants had a previous relation or interest conflict with the researchers.

The external validity refers to the capacity to repeat the same behaviour considering other participants. To minimise this threat must be considered the different methodologies and technologies applied by the software organisations. In this sense, the study considered different software organisations to cover different technologies and programming languages (e.g. industrial organisation with software development area).

Finally, regarding the conclusion validity, the obtained results cannot be generalised and must be viewed as preliminary results. This research aims to establish a base that could be used for other researchers to explore the development of ontologies for the coding phase in the software development process.

6 Discussion

Some relevant aspects related to this work that are important to discuss are: (i) the methodology followed to develop the presented ontology; (ii) research impact; (iii) the research implications; and (iv) practical implications:

(i) This research presents an adaptation of the original version of the Methontology Framework, where we include a new phase, *evaluation phase*. Originally the Methontology Framework considers the evaluation as a task, across all its phases (see Fig. 2), because strictly speaking it does not fully describe the process of evaluation. In this sense, the new phase brings several benefits such as a more comprehensive process to the software engineers, because it resemblance the software development life cycle. Our new phase describes the use of a reasoner to perform a verification of the ontology and CQ to do the validation task.

(ii) The followed methodology was useful to identify and classify the works found, the objective was to measure the progress of the work based on the Methontology development cycle, which allows us to find a gap in the code phase of software development and also helped us to compare our work with the literature. Current works mainly covered the specification as in the of the [28] where they present a list of guidelines for ontology development, they only identify terms and concepts which involves results only of the first phase of the followed methodology, other works such as in [50] where an ontology (taxonomy) is presented, however, this model was not translated into an ontology language. A few works such as [41, 44, 45] formalise the ontology, which means that the taxonomy or knowledge model domain was converted into an ontology language. In this sense, our work presents a validated ontology for the coding phase of the software development process, which a phase not addressed by any of the works that achieve the formalisation phase.

(iii) The main research implication of this work is the knowledge condensation through ontologies because previous research has modelled knowledge condensation empirically [13]. Therefore, our work established terms and concepts to work with ontologies in the coding phase of the software development cycle, so this ontology could help to condensate the knowledge by sharing terms and concepts of how to capture and classify the knowledge. Moreover, since the process of development and validation of ontology is described, other researchers or developers can follow our process to develop their ontology.

(iv) The practical implication is that the developers can use the presented ontology for the development of a knowledge *expertise* system, similarly to the entity relation model in databases. This system could manage the sources used by the developers to solve problems or doubts, so the ontology will serve as a reasoner about a developer *expertise* need, see the query in Fig. 8. The resources will be stored according to the terms and concepts established in the ontology.

7 Conclusions

In this work, we addressed the *expertise* location problem to reduce architectural knowledge vaporisation in the software development lifecycle, through an ontology obtained over interviews, a focus group and a survey form. The ontology links the artefacts (resources) with his creators (provider) and with the project where it was used. We identified works that reported ontologies that support different activities or phases in software development, however, these works do not follow a specific methodology to develop ontologies. On the other hand, we found some works that present an ontology based on a literature review, this means that neither those have a formal approach to developing an ontology nor an adequate evaluation. Consequently, some of the found ontologies are not formalised, it means that the model presented on the works found is not ready to take advantage in a system.

Our proposal presents a description of the development process of an ontology (formalised) including their evaluation, it means that the model is ready to implement in a system. The main contributions of this work are the support to *expertise* location through an ontology that can link the information about programmers or any member of a team with the resources used in a project. Therefore, the developer will be able to identify the provider or the source of an artefact, or developers with the knowledge to solve problems or doubts in a specific domain. It will mitigate the time wasted trying to find solutions to solve problems or doubts, consequently, the knowledge reuse will help to reduce the *architectural knowledge vaporisation*.

In this sense, another contribution is the approach of *knowledge condensation* concept is presented using ontologies. In the work of Borrego *et al.* [13] the *knowledge condensation* is presented, as well as a technological implementation of the same concept, where the knowledge classification is carried out through a mechanism of semi-fixed social tagging. It is referred as semi-fixed because developers could use any tag, but it must be related to a fixed meta-tag of a classification scheme (similar to a taxonomy), which was obtained through various empirical studies. Thus, knowledge classification is not based on a formal process to develop ontologies. In this paper, we present the first efforts of an implementation of the *knowledge condensation* concept, where the classification mechanism is based on an ontology formally obtained and validated. In consequence, using an ontology enable automated reasoning about *architectural knowledge* (artefacts and experts), reasoning with concepts and relationships similar to the way humans perceived interlinked concepts and a model that evolves with grow of data without affecting processes. As future work, this ontology will serve as the language to enable automated reasoning about high-level knowledge (*expertise*) according to the needs of developers trying to solve problems or doubts.

8 Acknowledgments

This work was partially supported by different scholarships granted by the Mexican Institutions of PRODEP and by the National Council of Science and Technology (whose acronym in Spanish is Conacyt) of Mexico, with scholarship number 616574 for the first author.

The authors are grateful to the participating companies: SASA, CECSO, IBM, SOA Software Factory, EMCORE, for the support provided to conduct the present study, and for their willingness to continue working with us in future projects.

9 References

[1] Lindvall, M., Rus, I.: 'Knowledge management for software Organizations', in *'Managing software engineering Knowledge'* (Springer Berlin Heidelberg, Berlin, Heidelberg), 2003, pp. 73–94

[2] Bjørnson, F.O., Dingsøyr, T.: 'Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used', *Inf. Softw. Technol.*, 2008, **50**, (11), pp. 1055–1068.

[3] van Vliet, H.: 'Knowledge sharing in software development', 2010, pp. 2–2.

[4] Becerra-Fernandez, I., Sabherwal, R.: *'Knowledge management: systems and processes'* (Routledge, london, UK, 2014).

[5] Jedlitschka, A., Ciolkowski, M., Denger, C., *et al.*: 'Relevant information sources for successful technology transfer: a survey using inspections as an example'. First Int. Symp. on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, Spain, 2007, pp. 31–40

[6] Sharif, K.Y., Buckley, J.: 'Observation of open source programmers' information seeking'. 2009 IEEE 17th Int. Conf. on Program Comprehension, Vancouver, Canada, 2009, pp. 307–308.

[7] Zagalsky, A., German, D.M., Storey, M.-A., *et al.*: 'How the R community creates and curates knowledge: an extended study of stack overflow and mailing lists', *Empir. Softw. Eng.*, 2018, **23**, (2), pp. 953–986.

[8] Voas, J.: 'A baker's dozen: 13 software engineering challenges', *IT Prof.*., 2007, **9**, (2), pp. 48–53

[9] Rupakheti, C.R., Hou, D.: 'Satisfying programmers' information needs in API-based Programming'. 2011 IEEE 19th Int. Conf. on Program Comprehension, Kingston, Canada, 2011, pp. 250–253

[10] Ericsson, K.A., Prietula, M.J., Cokely, E.T.: 'The making of an expert', *Harv. Bus. Rev.*, 2007, **52**, pp. 115–121.

[11] 'Manifesto for Agile Software Development.' [Online]. Available at: https://agilemanifesto.org/. [Accessed: 30-Apr-2019].

[12] Noordeloos, R., Manteli, C., Van Vliet, H.: 'From RUP to scrum in global software development: a case study'. 2012 IEEE Seventh Int. Conf. on Global Software Engineering, Porto Alegre, Brazil, 2012, pp. 31–40.

[13] Borrego, G., Morán, A.L., Palacio, R.R., *et al.*: 'Towards a reduction in architectural knowledge vaporization during agile global software development', *Inf. Softw. Technol.*, 2019, **112**, pp. 68–82

[14] Uikey, N., Suman, U., Ramani, A.K.: 'A documented approach in Agile software development', 2011

[15] Holz, H., Melnik, G., Schaaf, M.: 'Knowledge management for distributed agile processes: models, techniques, and infrastructure'. WET ICE 2003. Proc. Twelfth IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Linz, Austria, 2003, pp. 291–294

[16] Edwards, J.S.: 'Managing software engineers and their knowledge', in *'Managing software engineering knowledge'* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003), pp. 5–27

[17] McDonald, D.W., Ackerman, M.S.: 'just talk to me'. Proc. of the 1998 ACM Conf. on Computer supported cooperative work - CSCW '98, Seattle, WA, USA, 1998, pp. 315–324

[18] Borrego, G., Moran, A.L., Palacio, R., *et al.*: 'Understanding architectural knowledge sharing in AGSD teams: an Empirical Study'. 2016 IEEE 11th Int. Conf. on Global Software Engineering (ICGSE), Irvine, CA, USA, 2016, pp. 109–118.

[19] Babar, M. A.: 'Supporting the software architecture process with knowledge Management', in *'Software architecture knowledge management'* (Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 2009), pp. 69–86.

[20] Rus, I., Lindvall, M.: 'Knowledge management in software engineering', *IEEE Softw.*, 2002, **19**, (3), pp. 26–38

[21] Bosch, J.: 'Software architectue: the next step', in Oquendo, F., Warboys, B., (Eds.): *'Lecture notes in computer science'* (Springer Berlin Heidelberg, Germany, 2004), pp. 194–199

[22] Bin Ali, N.: 'Is effectiveness sufficient to choose an intervention?: considering resource use in empirical software engineering'. Int. Symp. on Empirical Software Engineering and Measurement, Ciudad Real, Spain, 2016, vol. 08-09-September-2016

[23] Gruber, T.R.: 'Toward principles for the design of ontologies', *Int. J. Hum. Comput. Stud.*, 1995, **43**, (5), pp. 907–928.

[24] Shiang, C.W., Tee, F.S., Halin, A.A., *et al.*: 'Ontology reuse for multiagent system development through pattern classification', *Softw. - Pract. Exp.*, 2018, **48**, (11), pp. 1923–1939

[25] Noy, N.F., Mcguinness, D.L.: 'Ontology development 101: aGuide to creating your first ontology', 2001

[26] Paredes-Valverde, M.A., Salas-Zárate, M.d.P., Colomo-Palacios, R., *et al.*: 'An ontology-based approach with which to assign human resources to software projects', *Sci. Comput. Program.*, 2018, **156**, pp. 90–103

[27] Fonseca, V.S., Barcellos, M.P., de Almeida Falbo, R.: 'An ontology-based approach for integrating tools supporting the software measurement process', *Sci. Comput. Program.*, 2017, **135**, pp. 20–44

[28] Serna, M.E., Serna, A.A.: 'Ontology for knowledge management in software maintenance', *Int. J. Inf. Manage.*, 2014, **34**, (5), pp. 704–710.

[29] Bhatia, M.P.S., Kumar, A., Beniwal, R.: 'Ontologies for software engineering: past, present and future', *Indian J. Sci. Technol.*, 2016, **9**, (9), pp. 1–16,

[30] de Souza, P.L., do Prado, A.F., de Souza, W.L., *et al.*: *'Improving Agile software development with domain ontologies'* (Springer, Cham, 2018), pp. 267–274

[31] Casanovas, P., Casellas, N., Tempich, C., *et al.*: 'OPJK and DILIGENT: ontology modeling in a distributed environment', *Artif. Intell. Law*, 2007, **15**, (2), pp. 171–186.

[32] Sure, Y., Staab, S., Studer, R.: 'on-To-knowledge methodology (OTKM)', in *'Handbook on ontologies'* (Springer, Berlin Heidelberg, 2004), pp. 117–132

[33] Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: 'The neon methodology for ontology engineering', in *'Ontology engineering in a networked world'* (Springer, Berlin Heidelberg, 2012), pp. 9–34

[34] Castillo-Barrera, F.E., Durán-Limón, H.A., Médina-Ramírez, C., *et al.*: 'A method for building ontology-based electronic document management systems for quality standards - the case study of the ISO/TS 16949:2002 automotive standard', *Appl. Intell.*, 2013, **38**, (1), pp. 99–113.

[35] Fernández-López, M., Gómez-Pérez, A., Juristo, N.: 'METHONTOLOGY: from ontological art towards ontological engineering'. Proc. Ontological Engineering AAAI-97 Spring Symp. Series, Palo Alto, CA, USA, 24-26 March 1997,

[36] Park, J., Sung, K., Moon, S.: 'Developing graduation screen ontology based on the METHONTOLOGY approach'. Proc. - 4th Int. Conf. on Networked Computing and Advanced Information Management, NCM 2008, Gyeongju, South Korea, 2008, vol. 2, pp. 375–380

[37] Nur Husna, M.H., Zakaria, Z.: 'The development of STIF in ontology based on the methontology Approach', *UTM Comput. Proc. Innov. Comput. Technol. Appl.*, 2017, **2**, pp. 1–7

[38] 'protégé.' [Online]. Available at: https://protege.stanford.edu/. [Accessed: 13-May-2019].

[39] Sommerville, I.: *'Software engineering'* (Pearson, London, UK, 2016, 10th edn.).

[40] Pressman, R.S.: *'Software engineering: a practitioner's approach'* (McGraw-Hill, New York, USA, 2014).

[41] Hovorushchenko, T., Pavlova, O.: 'Evaluating the software requirements specifications using ontology-based intelligent Agent'. 2018 IEEE 13th Int. Scientific and Technical Conf. on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2018, pp. 215–218

[42] Bhatia, M.P.S., Kumar, A., Beniwal, R.: 'Ontology based framework for detecting ambiguities in software requirements specification'. Int. Conf. on Computing for Sustainable Global Development (INDIACom), New, Delhi, India, 2016.

[43] Sitthithanasakul, S., Choosri, N.: 'Using ontology to enhance requirement engineering in agile software process'. 2016 10th Int. Conf. on Software, Knowledge, Information Management & Applications (SKIMA), Chengdu, China, 2016, pp. 181–186.

[44] Murtazina, M.S., Avdeenko, T.V.: 'An ontology-based approach to support for requirements traceability in Agile development', *Procedia Comput. Sci.*, 2019, **150**, pp. 628–635.

[45] Khatoon, A., Motla, Y.H., Azeem, M., *et al.*: 'Requirement change management for global software development using ontology'. 2013 IEEE 9th Int. Conf. on Emerging Technologies (ICET), Islamabad, Pakistan, 2013, pp. 1–6

[46] Rocha, R.G.C., Meira, S.: 'DSDK: an Ontology-based system to explore distributed software development Experiments'. 2012 IEEE Seventh Int. Conf. on Global Software Engineering Workshops, Porto Alegre, Brazil, 2012, pp. 73–75

[47] Vizcaíno, A., García, F., Caballero, I., *et al.*: 'Towards an ontology for global software development', *IET Softw.*, 2012, **6**, (3), p. 214

[48] Vizcaíno, A., García, F., Piattini, M., *et al.*: 'A validated ontology for global software development', *Comput. Stand. Interfaces*, 2016, **46**, pp. 66–78.

[49] Bhatia, B.M.P.S., Kumar, A., Rohit, : 'Ontology based framework for automatic software's documentation'. Int. Conf. on Computing for Sustainable Global Development, New Delhi, India, 2015.

[50] Hamdan, K., Khatib, H., Moses, J., *et al.*: 'A software cost ontology system for assisting estimation of software project effort for use with case-based Reasoning'. 2006 Innovations in Information Technology, Dubai, UAE, 2006, pp. 1–5.

[51] Adnan, M., Afzal, M.: 'Ontology based multiagent effort estimation system for scrum Agile method', *IEEE Access*, 2017, **5**, pp. 25993–26005.

[52] Wongthongtham, P., Chang, E., Dillon, T.S., *et al.*: 'Ontology-based multi-site software development methodology and tools', *J. Syst. Archit.*, 2006, **52**, (11), pp. 640–653.

[53] Wongthongtham, P., Chang, E., Dillon, T., *et al.*: 'Development of a software engineering ontology for multisite software development', *IEEE Trans. Knowl. Data Eng.*, 2009, **21**, (8), pp. 1205–1217.

[54] Marques, A.B., Carvalho, J.R., Rodrigues, R., *et al.*: 'An ontology for task allocation to teams in distributed software Development'. 2013 IEEE 8th Int. Conf. on Global Software Engineering, Bari, Italy, 2013, pp. 21–30.

[55] Valiente, M.-C., Garcia-Barriocanal, E., Sicilia, M.-A.: 'Applying ontology-based models for supporting integrated software development and IT service management processes', *IEEE Trans. Syst. Man, Cybern. Part C (Appl. Rev.)*, 2012, **42**, (1), pp. 61–74.

[56] Martinho, R., Varajão, J., Domingos, D.: 'Using the semantic web to define a language for modelling controlled flexibility in software processes', *IET Softw.*, 2010, **4**, (6), p. 396,

[57] Zhang, Y., Witte, R., Rilling, J., *et al.*: 'Ontological approach for the semantic recovery of traceability links between software artefacts', *IET Softw.*, 2008, **2**, (3), p. 185,

[58] Rocha, R., Araujo, A., Cordeiro, D., *et al.*: 'DKDOnto: an Ontology to support software development with distributed teams', *Procedia Comput. Sci.*, 2018, **126**, pp. 373–382.

[59] Suárez-Figueroa, M.C., Gómez-Pérez, A., Villazón-Terrazas, B.: 'How to write and use the ontology requirements specification Document'. Proc. of the Confederated Int. Conf.s, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part II, Vilamoura, Portugal, 2009, pp. 966–982.

[60] 'Swoogle.' [Online]. Available at: http://swoogle.umbc.edu/2006/. [Accessed: 11-Apr-2019]

[61] 'DAML Ontology Library.' [Online]. Available at: http://www.daml.org/ontologies/. [Accessed: 11-Apr-2019]

[62] 'ONKI Ontology Library Service.' [Online]. Available at: https://onki.fi/en/. [Accessed: 11-Apr-2019]

[63] 'Linked Open Vocabularies (LOV).' [Online]. Available at: https://lov.linkeddata.es/dataset/lov/. [Accessed: 11-Apr-2019]

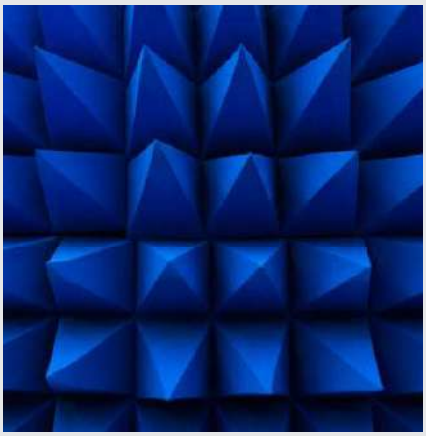
[64] Bezerra, C., Freitas, F., Santana, F.: 'Evaluating ontologies with competency Questions'. 2013 IEEE/WIC/ACM Int. Joint Conf.s on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Atlanta, GA, USA, 2013, pp. 284–285

[65] 'OWL Web Ontology Language Overview.' [Online]. Available at: https://www.w3.org/TR/owl-features/. [Accessed: 30-May-2019]

[66] Horridge, M., Patel-Schneider, P.F.: 'Manchester OWL Syntax', 2007. [Online]. Available at: https://www.w3.org/TR/owl2-manchester-syntax/. [Accessed: 06-Aug-2019].

[67] Barros, M.d.O., Neto, A.C.D.: 'Threats to validity in search-based software engineering empirical studies', *RelaTe-DIA 5.1*, 2011, **5**, (1), pp. 1–11

2018 AVIIES



AVANCES DE INVESTIGACIÓN EN INGENIERÍA EN EL ESTADO DE SONORA

Año 4, número 1
ISSN: 2448-7473

Responsable de la Edición del volumen
Dr. Ramón René Palacio Cinco

Colaboradores en la edición:
Dr. Mario Barceló Valenzuela
Dr. Alonso Pérez Soltero
Dr. Oscar Mario Rodríguez Elías
Dr. Guillermo Valencia Palomo
Dr. Joaquín Cortez González

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

AVANCES DE INVESTIGACIÓN EN INGENIERÍA EN EL ESTADO DE SONORA, Año 4 Núm 1, noviembre de 2018, es una revista anual, publicada y editada por el Tecnológico Nacional de México dependiente de la Secretaría de Educación Pública, a través del Instituto Tecnológico de Hermosillo, por la División de Estudios de Posgrado e Investigación, con domicilio en Arcos de Belén No. 79, piso 2, Colonia Centro, Delegación Cuauhtémoc, Ciudad de México, C.P. 06080, Tel. 5536017500, Correo electrónico: d_vinculacion@tecnm.mx. Editor Responsable: Dr. Oscar Mario Rodríguez Elias. Reserva de derechos al uso exclusivo No. 04- 2015-101310132700-203, con ISSN: 2448-7473, ambos otorgados por el Instituto Nacional del Derecho de Autor.

Responsables de la última actualización de este volumen: Dr. Ramón René Palacio Cinco, en colaboración con: Dr. Joaquín Cortez González, Dr. Mario Barceló Valenzuela, Dr. Alonso Pérez Soltero, Dr. Guillermo Valencia Palomo y Dr. Oscar Mario Rodríguez Elías, en las instalaciones del Instituto Tecnológico de Hermosillo, Ave. Tecnológico y Periférico Poniente SN C.P. 83170, Colonia Sahuaro, Hermosillo, Sonora, México. Fecha de término de impresión, 08 de Noviembre de 2018.

Su objetivo principal es difundir los avances en investigación a nivel posgrado y licenciatura en diversas áreas de la ingeniería, realizados durante el lapso de un año, en las instituciones participantes de educación superior del estado de Sonora.

Enlace de acceso: www.aviies.ith.mx

Avances de Investigación en Ingeniería en el Estado de Sonora

Año 4, Número 1

ISSN: 2448-7473

Responsable de la edición del volumen:
Dr. Ramón René Palacio Cinco

Colaboradores en la edición:
Dr. Mario Barceló Soltero
Dr. Alonso Pérez Soltero
Dr. Oscar Mario Rodríguez Elias
Dr. Guillermo Valencia Palomo
Dr. Joaquín Cortez González

Avances de Investigación en Ingeniería en el Estado de Sonora

Año 4, Número 1

ISSN: 2448-7473

Responsable de la edición del volumen:

Dr. Ramón René Palacio Cinco

Colaboradores en la edición:

Dr. Mario Barceló Soltero

Dr. Alonso Pérez Soltero

Dr. Oscar Mario Rodríguez Elias

Dr. Guillermo Valencia Palomo

Dr. Joaquín Cortez González

Maestría en Ingeniería Electrónica
Maestría en Ingeniería Industrial
y Maestría en Ciencias de la
Computación

Posgrado en
Ingeniería Industria

Maestría en Tecnologías de la
Información para los Negocios

Noviembre 2018 **ISSN: 2448-7473**

Avances de Investigación en Ingeniería en el Estado de Sonora,
Año 4, Número 1.
Responsable de la edición del volumen: Dr. Ramón René Palacio Cinco
Colaboradores en la edición: Dr. Mario Barceló Soltero, Dr. Alonso Pérez Soltero, Dr. Oscar Mario Rodríguez Elias, Dr. Guillermo Valencia Palomo, Dr. Joaquín Cortez González.

Posgrado en Ingeniería Industrial
División de Ingeniería
Universidad de Sonora
Maestría en Ingeniería Electrónica
Maestría en Ingeniería Industrial y
Maestría en Ciencias de la Computación
División de Estudios de Posgrado e Investigación
Instituto Tecnológico de Hermosillo
Maestría en Tecnologías de la Información para los Negocios
Instituto Tecnológico de Sonora

Índice de Contenido

A.- Resultados de Investigación

Computación y Tecnologías de Información

Análisis de la herramienta de TI para el apoyo a los deportistas de alto rendimiento con relación a su desempeño académico en el ITSON.

Felipe de Jesús Félix Hernández[1], Carlos Jesús Hinojosa Rodríguez [2]

[1] Instituto Tecnológico de Sonora, Campus Náinari,
Av Antonio Caso 2266, Colonia Villa ITSON, 85137 Cd Obregón, Sonora, México.
felix_4@hotmail.es
[2] Instituto Tecnológico de Sonora, Campus Navojoa,
Av Ramon Corona, Colonia ITSON, Navojoa , Sonora, México.
carlosjersus.hinojosa@gmail.com

Resumen. El presente artículo se centrará en el análisis de las distintas herramientas de TI con el propósito de reducir el alto índice de reprobación que actualmente se presenta en las Instituciones Educativas de Educación Superior en los estudiantes deportistas de alto rendimiento. Esta investigación se realizara en el Instituto Tecnológico de Sonora, con el departamento de deportes y los deportistas universitarios de alto rendimiento con la intención de encontrar alternativas de ayuda adecuada para gestionar la alternancia de esta doble actividad. Por ello se propone el análisis y comparación de distintas herramientas de TI.
Con la finalidad de facilitar su trayectoria escolar a través de un monitoreo que permita ver cuáles son las condiciones bajo las cuales se lleva a cabo el proceso de aprendizaje, se tomara como referencia la metodologia propuesta por Arguelles. Para ello se abordaran las etapas de dianostico y analisis, buscando tomar decisiones anticipadas que lograran contrarrestar esta problemática.

Palabras clave: Herramientas de TI, Instituciones de Educación Superior, deportistas de alto rendimiento, monitoreo, trayectorias escolares.

1 Introducción

Actualmente el proceso de formación universitaria de los deportistas de alto nivel se convierte muchas veces en un conflicto de interés, en el que se tienen que enfrentar a sus principales objetivos académicos y deportivos. Es por ello que existe una correlación

Félix Hernández FdJ, Hinojosa Rodriguez CJ (2018) Análisis de la herramienta de TI para el apoyo a los deportistas de alto rendimiento con relación a su desempeño académico en el ITSON. Avances de Investigación en Ingeniería en el Estado de Sonora 4 (1):56-63

ISSN 2007-9737

A Social Network to Increase Collaboration and Coordination in Distributed Teams

Aurora Vizcaíno[1], Pedro Garrido[1], Ramón R. Palacio[2], Alberto L. Morán[3], Mario Piattini[1]

[1] Universidad Castilla-La Mancha, Grupo de Investigación Alarcos, España

[2] Instituto Tecnológico de Sonora, Unidad Navojoa, Mexico

[3] Universidad Autónoma de Baja California, Facultad de Ciencias-Ensenada, Mexico

{aurora.vizcaino, mario.piattini}@uclm.es, pedro8853@hotmail.com, ramon.palacio@itson.edu.mx, alberto.moran@uabc.edu.mx

Abstract. Trust is one of the key factors involved in determining the success or failure of any project. However, achieving and maintaining trust in distributed projects when team members are geographically, temporally and culturally distant from each other is a considerable challenge. In this paper, we present Trusty, a tool designed to help develop trust in Virtual Teams. The tool is explained by using a schema of trustworthiness, and an indication of how the tool supports some features of these schema in order to foster the development of trust is therefore provided. Users have also evaluated the tool, and the results of this evaluation are presented here.

Keywords. Global software development, trustworthiness, virtual teams.

1 Introduction

The last few decades have witnessed a steady, irreversible trend towards the globalisation of business. Economic forces are relentlessly turning national markets into global markets and spawning new forms of competition and cooperation that reach across national boundaries. This change is having a profound impact on not only marketing and distribution, but also the way in which products are conceived, designed, constructed, tested, and delivered to customers [1].

Companies are therefore expanding globally, and are distributing their teams around the world by a variety of means such as acquisitions, partnerships, and outsourcing. As globalisation becomes more prevalent, many companies are evolving their approach and practices, and thus perhaps demonstrating the maturity of the distributed model. It is the age of Virtual Teams (VTs), in which members use technology to interact with one another across geographic, organisational, and other boundaries [2]. VTs can be composed of the best individuals for the task regardless of their physical or organisational location, thus enhancing the quality of decisions [3]. Furthermore, in order to attract and retain employees, and knowledge workers in particular, organisations are increasingly offering their employees remote working options [4]. Overall, VTs provide an effective structural mechanism with which to handle the increased travel, time, coordination, and costs associated with bringing together geographically, temporally, and functionally dispersed employees to work on a common task. Over the last decade, researchers have sought to understand the benefits and costs associated with VTs. Given this, there is now a burgeoning amount of literature on VTs that spans multiple disciplines [5].

Nevertheless, various challenges appear in VTs, one of which is a lack of trust that leads to other important consequences such as “poor socialisation and socio-cultural fit, absence of

conflict handling and lack of cognitive-based trust, increasing monitoring, inconsistency in work practices and both a decrease and unpredictability in communication" [6, 7]. Lack of trust can thus cause a decrease in productivity, quality and information exchange.

It is, however, difficult to build and foster trust by using an application, since the conditions associated with distribution are very demanding owing to the fact that most of the traditional sources of trust do not exist in networked conditions. Consequently, trust in networks may emerge occasionally, but maintaining and fostering it is particularly challenging [7-11].

Our awareness of this problem led us to study how this lack of trust could be avoided or decreased. Social Networking Sites (SNS), may be one alternative that can be used for this purpose, since they have the capacity to permit members of a virtual group to share experiences, exchange information and present themselves in real-time [10]. These features of SNS encouraged us to develop a tool based on the idea of a social network that helps to build trust among VT users. This tool is called Trusty.

The Trusty tool was therefore designed with the purpose of facilitating the fostering of trust among team members. The functionality of "Trusty" has consequently been aligned and presented according to the schema of trustworthiness proposed by [11]. Furthermore, in this paper we present the results of the mechanisms and information elements of Trusty as regards their trustworthiness, which were tested by 100 developers from 5 different cities in Mexico.

2 Background

In the literature, the term "trust" acquires various meanings according the context in which it appears. Trust is generally defined as a "positive characteristic leading to desirable behavior and outcomes". According to [12], it is therefore possible to find different types of trust, which are:

i) Personal or impersonal, including cognitive trust, which refers to beliefs about others' competence and reliability. This can lead individuals to engage in less self-protective actions and be more likely to take risks. This type also includes affective trust, which refers to what arises from emotional ties among group members that reflect beliefs about reciprocated care and concerns.
ii) Swift or fragile. Swift trust occurs when people obtain trust from previous settings in the present. This emerges in a work context and in a limited history of working together, diverse member skills, etc. Fragile trust is a positive trust that is vulnerable to opportunistic defections. It generally develops early in a team's life cycle [13].
iii) Positive or Negative. Even when positive trust is desirable, negative trust and distrust may emerge. Negative cognitive trust occurs when a trustor believes that a trustee will not fulfil commitments and does not have the necessary competencies and skills to make an effective contribution. Mistrust may therefore stem from the unknown and can change to positive trust if expectations are met or exceeded.

Trust building is important, but more important is the initial trust building, because it is a process in which the trustee's trustworthiness is evaluated and expectations are negotiated [14], such that if the expectations about a trustee are not clear and well set out from the beginning, subsequent efforts to achieve or maintain trust will be useless [13].

3 Schema of Trustworthiness

All of the above has led different researchers to make efforts to develop and maintain trust during virtual teamwork [4, 15-17] in which they have identified that external signals (reputation, roles, rules), and intrinsic factors (predisposition to trust), determine initial swift trust. Moreover, assessments of benevolence and the continued assessment of integrity determine trust during the final stages of work, signifying that external signals (reputation, roles and rules), and intrinsic factors (predisposition to trust), determine initial swift trust. An appreciation of ability and integrity (cognitive trust), also enables trust to be established when a team first begins to work together. Benevolence (affective trust) ,and the continued assessment of integrity similarly determine trust in the later stages [13].

Fig. 1. Model for the schema of trustworthiness proposed in [11]

Fig. 2. Trusty tool view

Fig. 3. Public profile information provided by trusty

The features of trust described previously are used by the authors of [11], to propose a method with which to improve the creation of interpersonal trust in a virtual team, the type of trust in which we are interested for this study.

The proposal from the aforementioned study will be used to show how the Trusty tool fosters interpersonal trust in VTs. The schema of trustworthiness, which has five main categories as is shown in Figure 1, is summarised in the following section.

4 Fostering Trust with Trusty

Trusty is a tool which was designed with the goal of fostering trust in VT's (see Figure 2). Trusty was developed to have the following main capabilities:

- To provide useful information about co-workers, focusing on easing the communication among team members.
- To provide mechanisms through which to share informal information in order to increase the friendship among members and, consequently, the team's spirit of trust.
- To provide mechanisms to support communication by means of a set of groupware tools.
- To provide mechanisms to support knowledge sharing.
- To provide mechanisms to support coordination by means of event creation and sharing.
- To provide reports on and statistical analyses of the social network supported by the tool in order to help project leaders to obtain feedback about members' interactions.

We have taken the schema of trustworthiness proposed by [11], as a reference model to explain how Trusty tool fosters trust during teamwork.

In this section we therefore describe how reliability can be perceived by a Trustor as regards the information elements that impact on the categories proposed in the schema of trustworthiness.

4.1 Communality

The first category that [11] considers important in order to foster trust is communality, which refers to the personal characteristics that the trustor has in common with the trustee. This can be any shared characteristic, like a similar goal that they wish to achieve, shared language use, common identity characteristics or shared values.

Trusty attempts to foster Communality by providing different types of information stated in three profiles: a public profile, a project group profile and a personal profile (see Figure 3A). These allow trustees to discover any characteristics that they may have in common with a particular trustee.

The *public profile* shows general information about stakeholders (trustee). It is therefore visible to all the people in the organisation in order to provide information that will allow them, for instance, to communicate with each other. The information shown in this profile is considered to be common (gender, nationality, native language and level of knowledge of foreign languages).

We considered that it was necessary to show information regarding gender (see Figure 3A) because some people feel more comfortable interacting with people of the same gender, or vice versa and sometimes it is difficult to know whether you are interacting with a man or a woman just by their name. A mistake of this nature may be offensive or embarrassing [18].

Furthermore, the language is very important for communality, since it can be a key factor (see Figure 3B). This is because the language will be the communication system that will allow the stakeholders to communicate and exchange their ideas [19-21].

It is thus important to know a trustee's level of knowledge of languages because the common language among stakeholders could increase the trust needed to start an interaction [22]. The objective of the *project profile* is to share information about those members who are working on the same project, which might make communication and coordination easier (see Figure 4).

Fig. 4. Project group profile information provided by trusty

Fig. 5. Personal profile information provided by trusty

This profile includes all the information in the public profile and also appends (see Figure 4A) project-related information such as the name of the project on which a person is working or has worked, their role in the project, current activities, forthcoming events (see Figure 4B), etc. Information concerning people skills (see Figure 4C) and place of work (see Figure 4D) can also be included. This information helps to locate where the other person is, as communicating with a colleague without knowing where that person is located may sometimes make one feel uncomfortable [23].

This information allows the trustor to discover features that s/he may have in common with the trustee, signifying that using information related to the type of project, role and knowledge can help to generate more willingness to interact [24].

The *personal profile* helps to share more private aspects, which is critical when attempting to foster trust. For instance, the culture a person is from may allow trust to be fostered among partners because culture plays a key role in the context of VTs [25], since it is clearly reasonable to believe that if you know more about a person, you might have more criteria to decide whether that person is trustworthy. Moreover, according to [26], how well people know each other has an impact on team spirit. This profile gives people the opportunity to share more information about themselves and to provide a channel for informal communication in VTs, with the objective of increasing mutual knowledge and helping to build trust [23]. The personal profile (see Figure 5) includes other data items that are specifically related to the person in order to encourage interpersonal interaction. This profile is only visible to people that have been previously accepted as “friends”. The importance of understanding cultural differences and the relevance this can have in the successful completion of projects should not be underestimated [25], since a trustor could feel more comfortable starting an interaction with a trustee from the same or a similar culture (see Figure 5A) [27]. In contrast, the interest information (see Figure 5B) provides data concerning personal preferences, such as hobbies, activities, etc. So, unlike other (social networks) tools, the personal profile of Trusty is oriented to establish a formal communication, and do it as smoothly as possible, among the team members providing information such as culture, hobbies, personal interests, etc.; one of the reasons for adding this type of information elements is for users to find their personal interest characteristics with their colleagues, to facilitate the starting of communication and to form their working community.

Trusty additionally includes the information element “contact by” which allows a trustee to indicate the means by which media s/he prefers to be contacted (see Figure 2). That is, when a trustor identifies that a possible trustee has chosen the same means of communication, this could encourage the trustor to contact him/her since they could interact by the same means in a comfortable manner.

4.2 Ability

In order to foster trust, it is important to know a trustee’s capabilities, determined by knowledge, skills and competences, which enable tasks to be performed within a specific domain.

The project group profile provides two sections in which abilities are shown: Information about the trustee’s roles and the project in which s/he is involved (Figure 6A) and type of experience with technology use (Figure 6B). The personal profile also provides more data about skills and knowledge [28], such as previous work experience (Figure 6C) and academic studies (Figure 6D). This information will allow the trustor to perceive a trustee’s capabilities in a rapid and explicit manner. This kind of information could be useful when assigning tasks, and more so when these tasks are critical to a project [29].

4.3 Benevolence

This category refers to the perceived level of courtesy and positive attitude a trustee displays towards the trustor. It includes the extent to which a person seems: willing to help, available, sharing, to have faith in intentions, receptive, kind, open, caring and committed. Controlling benevolence in a tool can be a challenge.

However, we explain how we believe that the different features of Trusty could help a trustor to detect the positive attitude towards collaboration

ISSN 2007-9737

that a particular person has (willingness to help, availability and sharing).

One important characteristic of Trusty is the existence of a mechanism that detects *availability* for contact [24], identifying the best moment at which to initiate communication with other users based on their personal preferences. To do this, user profiles in Trusty show information about the user's working hours, the time at his/her site, and the most important aspects of his/her current status regarding availability, his/her preferred time to be contacted, etc. It should be highlighted that users provide some of this information when they define their profiles (see Figure 7).

Trusty includes a mechanism that helps to choose the best moment at which to initiate communication with another user based on people's personal preferences.

To do this, each person provides, and his/her profile shows, information about their working hours, their current status regarding availability (see right-hand side of Trusty screens in Figure 2), the time people prefer to be contacted, etc. In addition, it has been shown that interruptions have a negative impact on task completion time [14, 30], decision-making [31, 32], and people's emotional states [33]. Interruptions may also result in prospective memory failure [30, 34], which refers to the fact that an individual may have a problem remembering what s/he has to do as regards a planned task (or in this case, the interrupted task). Moreover, in order to make this information clearer for the users, Trusty represents the user's status with a colour code similar to that of CWS [35]. This colour code is guided by the selective availability criteria [36], such as "*I am available only to people who are related to the task I am dealing with now and am not available to other people*".

Trusty does this by using different colours on the photo frame in the panel on the right of the screen in order to indicate whether or not it is an appropriate moment to start a synchronous interaction with the other person. There are five possible colours (blue, green, yellow, orange and red). The colour code for the photo frame, taking into account the setting of the current status and the time at the site with regard to the hours at which that person prefers to be contacted.

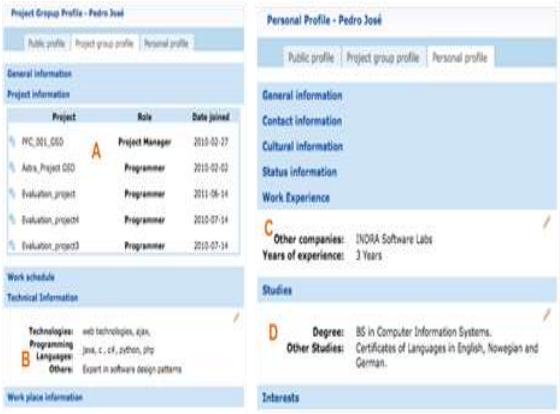


Fig. 6. Ability to view information using trusty

Fig. 7. Availability view

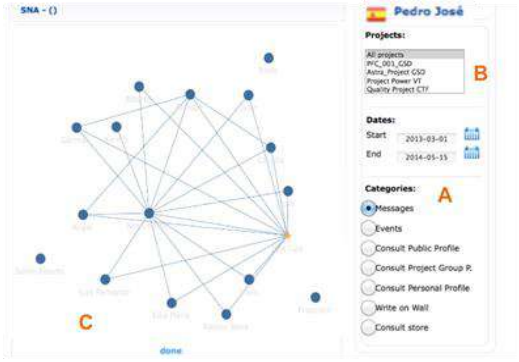


Fig. 8. Statistics (SNA) view

Table 1. Trusty versus continuous coordination tools

	Communality	**Ability**			**Benevolence**								**Internalized norms**							**Accountability**				
Tools		Knowledge	Competence	Skills	Willingness to help	Availability	Sharing	Faith in intentions	Receptively	Friendliness	Openness	Caring	Commitment	Integrity	Discretion	Honestly	Fairness	Loyalty	Reliability	Consistency	Self-confidence	Persistence	Responsibility
Palantir													X										X
Workspace Activity Viewer			X										X										
Ariadne						X																	X
World View	X		X			X																	
Trusty	X	X	X	X	X	X	X	X	X		X		X										X

Table 2. Trusty versus enterprise social networks

	Communality	**Ability**				**Benevolence**							**Internalized norms**							**Accountability**				
Tools		Knowledge	Competence	Skills	Willingness to help	Availability	Sharing	Faith in intentions	Receptively	Friendliness	Openness	Caring	Commitment	Integrity	Discretion	Honestly	Fairness	Loyalty	Reliability	Consistency	Self-confidence	Persistence	Responsibility
Yammer	X					X	X																
Zyncro	X						X						X										
Kudos			X																				
Facebook	X						X																
Twitter							X																
LinkedIn	X	X																					
IBM SB	X						X						X										
Trusty	X	X	X	X	X	X	X	X			X		X										X

Another of the capabilities that we wished to include in the design of Trusty was that of obtaining information about the usage of the tool by team members. This information may, for example, be useful in detecting that a particular person is undergoing message overload or the lack of interaction between certain team members. The algorithm that makes this possible is based on SNA [37]. The information is shown as a graph on which nodes represent Trusty users (see Figure 8C). This statistical mechanism is accessible to project managers and system administrators. The tool can be used to analyse various aspects of interaction on the social network, including message traffic, event publication, wall usage, profile visits, and knowledge repository usage (see Figure 8A). We therefore believe that this tool helps to increase positive leadership, team spirit and enthusiasm because it helps, for instance, to detect a particular worker's overload or whether there is a person who might have communication problems since s/he does not use any communication mechanisms (see Figure 8B). It is also possible to discover benevolence by analysing which people are contacted most often and whether or not they respond. This allows a trustor to "infer" whether the trustee is an open, kind or receptive person.

For instance, in Figure 8C we can see that there are several isolated nodes (Pablo, Jaime Alberto and Francisco), and this may be a sign of a problem, since as all these people are working on the same project it is logical to believe that all of them have to use some type of communication. When the project manager detects this situation by looking at the graph, s/he should attempt to find out why this situation has occurred. It might be that these people are on holiday at that time, and it is not therefore a problem. However, it could be a problem if these people do not communicate because they are shy or have problems understanding the messages, etc. In contrast, the node tagged as "Ana Lourdes" shows a lot of interaction with several members, and the project manager could therefore attempt to find out whether this person is overloaded or is an expert in a topic and is helping other teams' members.

Social Networks Analysis (SNA) [38], permits us to infer that a trustee has the characteristic of openness by viewing the items shared and the interest taken as regards interacting, even if s/he constantly responds to requests to interact [39]. The trustor can also infer whether a trustee is committed [40] and is interested in what is happening around the trustor, i.e. whether the trustee constantly participates on the trustor' wall. Moreover, when a trustee provides his/her availability schedule, a commitment indicator is shown.

According to literature, SNS is a good method with which to build trust in virtual teams. Furthermore, SNA can be used to obtain different information about team members, which might help to predict or detect possible problems in virtual teams, such as people who are isolated or overloaded, or a lack of communication among those that work in coupling tasks.

4.4 Internalised Norms

This category refers to the intrinsic moral norms a trustee uses to guard his/her actions. These differ from benevolence in that they are directed towards others in general, rather than toward a specific trustor. This includes the extent to which a person seems to have: integrity, discretion, honesty, fairness and loyalty [11].

The internalised norms are not potentiated with the tool, as we believe that they are very particular aspects of people's personalities. They have not therefore been considered when designing Trusty. However, Language Analysis regarding how a trustee uses the chat and walls could serve to infer some people's values. This language issue is not, however, within the scope of Trusty.

4.5 Accountability

This is the last category of the schema (see Figure 1) and refers to the degree to which a person is liable and accountable for his/her acts and meets the expectations of another person. It includes the extent to which a person seems to be: reliable, consistent, self- confident, persistent and responsible.

Trusty provides a list per project showing in which projects the trustor is involved. The items of information obtained from this list are project name, date joined, role, start date and completion of the project (responsibility). This kind of information makes it possible to know the workloads that teamwork members have accepted [41], and a trustor can therefore consult this information in order to see what responsibilities a person has and whether that person tends to meet deadlines.

5 Differences between this Social Application and Others

Several applications can support trust building in VT's. Table 1 shows a comparison between "Trusty" and various other continuous coordination tools. This comparison was performed according to how this tool fulfils the schema of trustworthiness proposed by [11]. A brief description of these tools is presented as follows:

- Palantír [42]: This application fosters benevolence towards the other team members, since it is possible to know which member has edited a module (commitment) and whether the task was completed (responsibility).

Table 3. Factors and internal consistency

#	ITEMS	Factors				
		1	**2**	**3**	**4**	**5**
1	The information that Trusty distributes forms part of my work activities.	.657				
2	The information that Trusty shows is in accordance with my communication needs at work.	.723				
3	Trusty's information elements could help me to resolve any doubts I may have about my colleagues' experience.	.537				
4	Trusty shows different information profiles that could help me to identify a colleague with similar interests to my own.	.445				
5	The Trusty Project Group Profile shows information about a colleague's software development skills		.682			
6	I would be prepared to use Trusty to obtain information about my colleagues' expertise		.737			
7	Trusty allows me to analyse a colleague's level of interactions with the work group		.521			
8	Trusty provides information about colleagues in a clear way		.731			
9	A colleague's availability mechanism is appropriate as regards determining the best moment at which to contact me			.574		
10	The mechanism used to determine the best moment at which to contact a colleague is appropriate.			.551		
11	The assistance that I receive from the colour code in order to determine a colleague's state of availability is easy to understand.			.674		
12	Using Trusty to communicate with my colleagues is appropriate and useful.			.671		
13	The information provided about a colleague is sufficient for me to be able to contact him/her.			.678		
14	I shall recommend Trusty to my colleagues.				.538	
15	If anyone asks me about the Trusty system, I shall recommend it to them.				.496	
16	I shall encourage my colleagues to use the different services provided by Trusty.				.686	
17	If my organisation adopts Trusty, I shall use it to communicate with my colleagues				.574	
18	The personal information included in Trusty does not have a negative effect on me.				.727	
19	When using Trusty, it is easy to navigate and discover all that I need to know about my colleagues.				.459	
20	All the information provided by Trusty is supported in the software development work activities.				.438	
21	The image projected as regards the information provided by Trusty is one of integrity and good values to communicate with colleagues					.635
22	I can be sure that the use of my personal information will be managed with discretion and not made public, but will only be used by the organisation.					.607
23	The information provided by Trusty is truthful and verifiable.					.640
24	The information in Trusty can keep me informed about a colleague's workload.					.553
25	I consider that I could become skilled in the use of Trusty in a short amount of time.					.551
26	Trusty is able to provide me with information about a colleague's project commitments.					.552
27	I consider that the information that Trusty distributes is consistent with the communication among colleagues in Software Development					.690
		Accumulated variance =56.87				
		Cronbach's Alpha = 0.917				

- Workspace Activity Viewer [43]: This application helps to create more accurate expectations (commitment), since it illustrates each member's prior performance (competence).
- Ariadne [30]: This application permits team members to monitor themselves (availability). It also provides an interactive analysis, which permits the project manager to adjust team members' tasks (responsibility).
- World View [30]: It uses intuitive visualisations to explain the team members' status by identifying relevant tasks (competence), irrelevant tasks (communality), and dependences (commitment).

As the results in Table 1 show, Trusty is the most complete application as regards fulfilling the schema of trustworthiness.

On the other hand, in Table 2 is shown a comparison between Trusty and social networks. It is important to highlight that the social networks selected have been promoted for use in companies. A description of these social networks is presented as follows:

- Yammer [44]: This social network includes microblogging, private chats, shared workspaces (availability) and document exchange (sharing).
- Zyncro [32]: This social network was designed to allow employees to recognize each other, which promotes engagement (commitment) with the enterprise.
- Kudos [45]: It is a microblogging application, which includes an employee recognition program and a corporate social network designed to engage the enterprise team with enhanced communication, collaboration, appreciation, recognition, and rewards (competence).
- Facebook [46]: The main features are sharing and communication among contacts considered as "friends" (sharing). This utility also permanently shows its members' public profiles (communality), signifying that it is possible to access personal data.
- Twitter [47]: Users can describe an actual situation or discuss a specific topic (sharing). These comments can be followed by users, thus allowing them to keep up to date with their topics of interest.
- LinkedIn [48]: It is thus possible to contact professional colleagues or old schoolmates. This network also makes it possible to become known in the professional field in order to find a job (communality).
- IBM Social Business (SB) [49]: This Social Business can help an organization extend customer relationships, generate new ideas faster (sharing), identify expertise (communally) and enable a more effective workforce (commitment).
- Table 2 shows Trusty as the social network that provides the most features to help develop trust. The fact that the SNS Analysis is included provides it with an important competitiveness and advantage over the other social networks, as important information can be obtained from these analyses. The differences shown in this section, with respect to Trusty's characteristics against other tools, were possible to determine by means of the factors of the Rusman Schema, since its factors helped us to make an analysis centered on the characteristics of Trustworthiness. With this it was possible to identify the shortcomings of the tools in terms of these factors and from there it could be possible to propose more suitable designs for the promotion of Trustworthiness among different users of an organization or virtual community.

6 Evaluation

The objective of this evaluation was to perceive the trustworthiness of Trusty by analysing users' opinions with regard to the performance of its mechanisms and services.

6.1 Design of the Study

A scenario was therefore designed whose objective was to test all the Trusty options, signifying that the users had to carry out all the activities indicated in the scenario. In order to test the trustworthiness of the application, we designed

a questionnaire on the basis of the schema of trustworthiness proposed by [11].

The decision was made to first carry out a pilot evaluation in order to test whether the scenario was as complete as possible (all the main functionalities were dealt with) and that the questionnaire was easy to understand. Two experts in Software Development created the activities and answered the questionnaire. They detected various limitations in the tool when it was used with Mac and they also suggested the addition of more activities in the evaluation scenario. Trusty was therefore improved and the proposed activities were added.

6.2 Subjects

The participants were 100 workers from different companies in five different Mexican cities. All of them were participating or had participated in Software projects. Their average age was 32 years old, and they all had at least three (3) years of experience in Software Development. All of them had Bachelor's degrees (BSc) in computer science or similar and eight (8) had Master's degrees (MSc) in computer science. They had different roles, i.e. there were 10 project managers, 2 testers, 25 programmers, 30 analysts, and one researcher. The remaining 33 respondents had played several roles, including programmer, analyst, tester or project manager. Each person was a member of a different software development enterprise in different geographical locations and they carried out the evaluation activities in their own workplaces.

6.3 Materials

This section describes the different materials used:

Scenario Document: This document described a set of scenarios for the fifteen activities that users have to perform in order to try out all the basic features of the tool.

Questionnaire regarding the tool: The questionnaire used to measure trustworthiness contained 27 questions quantified using a Likert scale of 1 (strongly disagree) to 5 (strongly agree). The average time needed to respond to the questionnaire was 15 minutes. Before responding to the questionnaire the participants were asked to state their years of experience in software development, their age, highest qualifications and the role they played in the organisation. This questionnaires was designed by using the schema of trustworthiness proposed by [11] to create an initial set of 50 questions to which the aforementioned people would respond. This preliminary format then was presented to a group of experts (psychologists and software engineers) in order for them to evaluate it. The analysis carried out allowed us to select the 32 questions contained in the first version of the trustworthiness questionnaire.

The concurrent validity of the questionnaire was obtained by means of contrasted groups obtained using the t test for independent samples, with the aim of identifying the questions that would show which participants had obtained a low mark as regards their perception of trustworthiness, and which had obtained high marks. We discovered that the total number of questions had p values of less than 0.05, i.e. all of them were discriminatory and were sensitive as regards identifying low and high marks. We next developed a frequency analysis of the questions in order to eliminate those that were most biased and had an asymmetric distribution, thus reducing the number of questions in the questionnaire to 27, which were then subjected to an exploratory factorial analysis using orthogonal rotation techniques in which the saturation point was 0.40.

This initially showed seven factors, five of which contained three or more questions. Those factors containing less than three questions were eliminated, leaving us with five factors. This factorial structure of 27 questions proved to be the most psychometrically appropriate and consistent, and was as follows:

- Factor 1: Communality (4 items).
- Factor 2: Ability (4 items).
- Factor 3: Benevolence (5 items).
- Factor 4: Internalized norms (7 items).
- Factor 5: Accountability (7 items).

The five factors, along with their respective questions, accumulated variance and Cronbach's alpha are shown in Table 3.

Table 4. Communality results

ITEMS	Mean (std dev.)
1. The information that Trusty distributes forms a part of my work activities.	3.72 (0.780)
2. The information that Trusty shows is in accordance with my communication needs at work.	3.67 (0.753)
3. Trusty's information elements could help me to resolve any doubts I may have about my colleagues' expertise.	3.66 (0.879)
4. Trusty shows different information profiles that could help me to identify a colleague with characteristics that are similar to my own.	3.76 (0.698)
Total	**3.70 (0.046)**

Table 5. Ability results

ITEMS	Mean (std dev.)
5. The Trusty Project Group profile shows information about a colleague's software development skills.	3.10 (0.870)
6. I would be prepared to use Trusty to obtain information about a colleagues' expertise.	2.74 (1.088)
7. Trusty allows me to analyse a colleague's level of interactions with the work group.	3.68 (0.634)
8. Trusty provides information about colleagues in a clear way.	3.68 (0.764)
Total	**3.30 (0.463)**

Table 6. Benevolence results

ITEMS	Mean (std dev.)
9. A colleague's availability mechanism is appropriate as regards determining the best moment at which to contact me.	3.80 (0.620)
10. The mechanism used to determine the best moment at which to contact a colleague is appropriate.	3.60 (0.696)
11. The assistance that I receive from the colour code in order to determine a colleague's state of availability is easy to understand.	3.84 (0.581)
12. Using Trusty to communicate with my colleagues is appropriate and useful.	3.86 (0.697)
13. The information provided about a colleague is sufficient for me to be able to contact him/her.	3.57 (0.807)
Total	**3.73 (0.138)**

The questionnaire as a whole obtained an internal consistence of α=0.917.

6.4 Procedure

Three activities were necessary for this evaluation, which were:

i) Initial Meeting. The participants were introduced to the study and were provided with the Trusty tool and its user manual.
ii) Trusty Activities. They were asked to perform the following activities with the tool, and they had one week to carry out the tasks:

- Update their general information.
- Update the profile of a project group.
- Update their personal profile.
- Perform searches to locate a user ("Thomas").
- Locate the partners in the projects in which they were also involved, and identify their nationalities.
- Locate and identify friends' hobbies.
- Post a message.
- See next month's events and their rates.
- See posts.
- Create a message.
- Send a Chat message.
- Use the Private Message chat application.
- See the files in the "Documentation" repository.
- Consult the amount of interactions in a user profile.
- Create a repository and upload a file

iii) On-exit survey. Finally, we asked the participants to fill in a questionnaire evaluating the trustworthiness of the System.

6.5 Limitations

The experiment described in this section and the methods used in order to evaluate it might have several weaknesses. The influence that these weaknesses may have had on the results is explained as follows: A) The results are focused on the participants' opinions and we do not therefore know whether being exposed to the system changed their perception of the technology. These results are restricted to a group of developers who work in geographic locations in Mexico, and it will therefore be difficult to replicate the results. B) Finally, this study is an exploratory work whose reach is focused on the trustworthiness of the use of Trusty in Software Development work environments.

6.6 Results and Discussion

The evaluation of Trusty was performed in collaboration with enterprises working in Global Software Development (GSD). The objective of this evaluation was to perceive the trustworthiness of Trusty by analysing users' opinions as regards the performance of its mechanisms and services. A scenario was therefore designed whose objective was to test all the options of Trusty, signifying that the users had to carry out all the activities indicated in the scenario, after which we analysed the participants' responses to the questions. The questionnaire was quantified using a Likert scale of 1 (strongly disagree) to 5 (strongly agree).

In the case of testing **Trusty's Communality**, the mean communality score that users gave to Trusty was 3.70 (s.d.= 0.046), as is shown in Table 5. We should state that the developers considered that the information distributed by Trusty is appropriate for DSD activities (mean = 3.72; s.d.= 0.780) and that they also considered that Trusty provides useful information with which to identify a colleague's characteristics.

However, although Trusty provides communality with adequate support, the mean obtained would have been higher if more detailed information elements that would enable the trustor to identify personal characteristics that s/he has in common with the trustee had been provided (e.g. types of projects on which they have participated or a link to their personal network.

In the case of testing **Trusty's Ability**, the mean Ability score that users gave to Trusty was 3.30 (s.d.= 0.463), as is shown in Table 5. According to the scale in the questionnaire, the developers considered that the information provided by Trusty is insufficient. This is evident if we observe the mean score obtained by Item 6, which was evaluated with a low mark (mean =2.74;

Table 7. Internalized norms results

ITEMS	Mean (std dev.)
14. I shall recommend Trusty to my colleagues.	3.90 (0.732)
15. If anyone asks me about the Trusty system, I shall recommend it to them.	4.00 (0.804)
16. I shall encourage my colleagues to use the different services provided by Trusty.	3.71 (0.902)
17. If my organisation adopts Trusty, I shall use it to communicate with my colleagues.	3.41 (0.889)
18. The personal information included in Trusty does not have a negative effect on me.	3.56 (0.935)
19. When using Trusty, it is easy to navigate and discover all that I need to know about my colleagues.	3.94 (0.694)
20. All the information provided by Trusty is supported in the software development work activities.	4.07 (0.590)
Total	**3.80 (0.245)**

Table 8. Accountability results

ITEMS	Mean (std dev.)
21. The image projected as regards the information provided by Trusty is one of integrity and good values to communicate with colleagues.	3.89 (0.665)
22. I can be sure that the use of my personal information will be managed with discretion and not made public, but will only be used by the organisation.	3.69 (0.748)
23. The information provided by Trusty is truthful and verifiable.	3.62 (0.838)
24. The information provided by Trusty can keep me informed about a colleague's workload.	3.66 (0.728)
25. I consider that I could become skilled in the use of Trusty in a short amount of time.	3.58 (0.755)
26. Trusty is able to provide me with information about a colleague's project commitments.	3.27 (1.004)
27. I consider that the information that Trusty distributes is consistent with the communication among colleagues in Software Development.	3.90 (0.732)
Total	**3.66 (0.213)**

s.d.=1.088), since the participants considered that they were not given sufficient information about their colleagues' skills. What is more, the information provided about the Project Group profile was not sufficient as regards their colleagues' development skills or capabilities.

In the case of testing **Trusty's Benevolence,** the mean benevolence score that users gave to Trusty was 3.73 (s.d.= 0.138), as is shown in Table 6. The participants considered that Trusty provides elements that allow them to perceive their colleagues' level of availability and willingness to

help, as is evidenced by Item 12 (mean 3.86; s.d.=0.697). In this case they perceive that the information provided by Trusty is useful for them as regards contacting their colleagues at appropriate moments. We should also mention that, with regard to Item 11, they found that the colour code provided by Trusty in order to identify a colleague's availability is easy to use and understand (mean=3.84; s.d.=0.581).

In the case of testing **Trusty's Internalised norms**, the mean Internalised norms score that users gave to Trusty was 3.80 (s.d.= 0.245), as is shown in Table 7. The participants considered that Trusty tool promotes activities in the Software Development work environment (mean=4.07; s.d.=0.590). They were also of the opinion that Trusty helped them to find information about their colleagues (mean=3.94; s.d.=0.694), thus promoting communication by means of different services (mean=3.71; s.d.=0.902), and signifying that they would recommend the tool to their colleagues (mean=4.00; 0.804). In general, the participants considered that Trusty provides information with which to find colleagues and that this information is used only to support work activities.

In the case of testing **Trusty's Accountability**, the mean Accountability score that users gave to Trusty was 3.66 (s.d.= 0.213), as is shown in Table 8. Trusty tool was considered to promote integrity and good values with the aim of communicating with colleagues (mean=3.89; s.d.=0.665), whilst respecting the discretion of the organisation of the information (mean=3.69; s.d.= 0.748). However, despite being a tool with which to exchange personal and professional information, Trusty was not considered sufficient as regards providing information about a colleague's forthcoming engagements (mean=3.27; s.d.=1.004).

The results obtained from the questionnaire do indicate that Trusty provides a suitable level of trustworthiness among software developers. However, an important adjustment should be made to it as regards Ability, since the mean scores obtained for the responses tended towards the neutral part of the scale (mean=3.30/5). In the case of the remaining dimensions, the participants tended to agree that the tool was useful, although Trusty should be adjusted in order to facilitate trustworthiness towards colleagues in an organisation, and it will therefore be necessary to include information elements or mechanisms that will enable trustworthiness towards colleagues to be enriched.

7 Conclusions and Future Work

In this paper we have described some of the challenges of VTs. Lack of trust is one of the challenges that also affects communication, coordination and control. In order to decrease these problems, we have developed Trusty, a tool that has been designed to help to develop trust among team members and also to make communication, coordination, and control easier. Trusty has been explained by following the schema of trustworthiness proposed by [11], and showing how Trusty covers most of the features included in this schema.

Moreover, Trusty was tested by means of an evaluation at different software companies whose team members worked with geographically distributed co-workers. The results obtained have provided us with some insights into how Trusty was perceived by workers as regards its trustworthiness. These results provide evidence that users tend to agree that Trusty fosters elements related to Communality, Benevolence, Internalized norms and Accountability but that the information about Ability is not sufficient. It will be necessary to continue working on these aspects as there is still room for improvement.

Trustworthiness was measured by creating a questionnaire based on the schema of [11], which obtained a high internal consistency (α=0.917), and we therefore consider that the questionnaire is both reliable and valid for the purposes of this measurement. This questionnaire can be used to measure the level of confidence fomented in the work group via the use of communication and/or coordination tools, using the information elements that are distributed with colleagues' contextual, personal and professional information as a starting point.

The results obtained have provided us with information that will allow us to identify those elements of the Rusman schema used in Trusty that were perceived to be the weakest. In this case, Ability was perceived to be the lowest, and we are

therefore contemplating a modified version of Trusty that will permit access to more detailed information as regards the information elements that Trusty currently provides, such that if the trustor requires more information about a trustee's skills, it will be possible to obtain it.

To conclude, we would like to state that Trusty could be used by any company or organisation whose teams are distributed throughout the world owing to a variety of collaboration strategies such as acquisitions, partnerships, and outsourcing. We believe that the tool will be very useful, principally in the first steps of collaboration during which people do not know each other and communication and collaboration among team members is important.

Trusty could also be useful in academic settings, since there is a strong tendency to collaborate on projects with people from other countries. When preparing a European project, it is advisable to create a multinational consortium in which not all the researchers have previous experience of working together, and Trusty could be a perfect means to start this collaboration and to help to develop trust and team spirit. Moreover, Trusty could help researchers to discover which person is the most suitable to ask for help when performing a particular task.

Acknowledgements

This work has been funded by the GINSENG project (Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional FEDER, TIN2015-70259-C2-1-R), by GEODAS-BC project (Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional FEDER, TIN2012-37493-C03-01), and by the LPS-BIGGER project: Línea de productos Software para BiG Data a partir de aplicaciones innovadores en entornos reales (Ref.: UCTR150175), is framed under the Strategic Program CIEN, and it is co-funded by "Centro para el Desarrollo Tecnológico Industrial (CDTI)", and "Fondo Europeo de Desarrollo Regional (FEDER), and GLOBALIA (PEII-2014-038-P), Consejería de Educación y Ciencia, Junta de Comunidades de Castilla-La Mancha.

References

1. **Damian, D. & Moitra, D. (2006).** Introduction: Global Software Development: How Far Have We Come?. *IEEE Software*, Vol. 23, No. 5, pp. 17–19.
2. **Gibson, C.B. & Cohen, S.G. (2003).** Virtual teams that work. Creating condition for virtual team effectiveness, *Personnel Psychology*, Vol. 57, No. 1, pp. 243–246.
3. **Lipnack, J. & Stamps, J. (1999).** Virtual teams: The new way to work. *Strategy & Leadership*, Vol. 27, No. 1, pp. 14–19. DOI: 10.1108/eb054625.
4. **Cascio, W.F. (2000).** Managing a Virtual Workplace. *Academy of Management Executive*, Vol. 14, No. 3, pp. 81–91, DOI:10.5465/ame. 2000.4468068.
5. **Martins, L.L., Gilson, L.L., & Maynard, M.T. (2004).** Virtual Teams: What Do We Know and Where Do We Go From Here?. *Journal of Management*, Vol. 30, No.6, pp. 805–835, DOI: 10.1016/j.jm.2004.05.002.
6. **Moe, N.B. & Šmite, D. (2008).** Understanding a lack of trust in Global Software Teams: a multiple-case study. *Softw. Process*, Vol. 13, No. 3, pp. 217-231, DOI: 10.1002/spip.378.
7. **Ali-Babar, M., Verner, J.M., & Nguyen, P.T. (2007).** Establishing and maintaining trust in software outsourcing relationships: An empirical investigation. *Journal of Systems and Software*, Vol. 80, No. 9, pp. 1438–1449, DOI: 10.1016/j.jss. 2006.10.038.
8. **McNab, A.L., Basoglu, K.A., Sarker, S., & Yanjun, Y. (2012).** Evolution of cognitive trust in distributed software development teams: a punctuated equilibrium model. *Electronic Markets*, Vol. 22, No.1, pp. 21–36. DOI:10.1007/s12525-011-0081-z.
9. **Al-Ani, B., Wilensky, H., Redmiles, D., & Simmons, E. (2011).** An Understanding of the Role of Trust in Knowledge Seeking and Acceptance Practices in Distributed Development Teams. *Proc. 6th IEEE International Conference on Global Software Engineering (ICGSE)*, pp. 25–34, DOI: 10.1109/ICGSE.2011.25.
10. **Colombo, G., Whitaker, R.M., & Allen, S.M. (2008).** Cooperation in Social Networks of Trust. *Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*. pp. 78–83. DOI:10.1109/ SASOW.2008.39.
11. **Rusman, E., Bruggen, J.V., & Koper, R. (2010)**. Fostering trust in virtual project teams: Towards a design framework grounded in a TrustWorthiness ANtecedents (TWAN) schema. *International*

Journal of Human-computer Studies, Vol. 68, No. 11, pp. 834–850. DOI: 10.1016/j.ijhcs.2010.07.003.

12. **Al-Ani, B. & Redmiles, D. (2009).** Trust in Distributed Teams: Support through Continuous Coordination. IEEE *Software*, Vol. 26, No. 6, pp. 35– 40. DOI: 10.1109/MS.2009.192.
13. **Greenberg, P.S., Greenberg, R.H., & Antonucci, Y.L. (2007).** Creating and sustaining trust in virtual teams. *Business Horizons*, Vol. 50, No. 4, pp. 325–333. DOI: 10.1016/j.bushor.2007.02.005.
14. **Jalali, S., Gencel, C., & Šmite, D. (2010).** Trust dynamics in global software engineering. *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Bolzano-Bozen*, pp. 1–9. DOI:10.1145/1852786.1852817.
15. **Anawati, D., & Craig, A. (2006).** Behavioral Adaptation Within Cross-Cultural Virtual Teams. *IEEE Transactions of proffessional communicacion pc*, Vol. 49, No. 1, pp. 44–56. DOI: 10.1109/TPC.2006.870459.
16. **Bavec, C. (2004)**. Trust - The Basis of a Virtual Organization. *Organizacija,* Vol. 37, No.10, pp. 594–595.
17. **Blanchard, A. & Markus, M. (2002).** Sense of Virtual Community-Maintaining the Experience of Belonging. *Proc. Sense of Virtual Community-Maintaining the Experience of Belonging (HICSS),* pp. 1–10. DOI: 10.1109/HICSS.2002.994449.
18. **Burleson, B.R. (2003)**. The experience and effects of emotional support: What the study of cultural and gender differences can tell us about close relationships, emotion, and interpersonal communication. *Personal Relationships*, Vol. 10, No. 1, pp. 1–23. DOI: 10.1111/1475-6811.00033.
19. **Aranda, G., Vizcaíno, A., & Piattini, M. (2010).** A framework to improve communication during the requirements elicitation process in GSD projects. *Requirements Engineering*, Vol. 15, No. 4, pp. 397–417. DOI: 10.1007/s00766-010-0105-9.
20. **Cataldo, M. & Herbsleb, J.D. (2008).** Communication patterns in geographically distributed software development and engineers' contributions to the development effort. *Proceedings international workshop on Cooperative and human aspects of software engineering*, pp. 25–28. DOI: 10.1145/1370114.1370121.
21. **Noll, J., Beecham, S., & Richardson, I. (2011)**. Global software development and collaboration: barriers and solutions. (*ACM),* Vol. 1, No. 3, pp. 66–78. DOI: 10.1145/1835428.1835445.
22. **Calefato, F., Lanubile, F., & Minervini, P. (2010)**. Can Real-Time Machine Translation Overcome Language Barriers in Distributed Requirements Engineering?. *5th IEEE International Conference on Global Software Engineering, (ICGSE),* pp. 257–264. DOI: 10.1109/ICGSE.2010.37.
23. **Aranda, G., Vizcaíno, A., Hernández, J., Palacio, R., Morán, A., Vivacqua, A., Gutwin, C., & Borges, M. (2011).** Trusty: A Tool to Improve Communication and Collaboration in DSD. *Collaboration and Technology, Vol. 6969*, pp. 224–231. DOI: 10.1007/978-3-642-23801-7_18.
24. **Ye, Y. (2006)**. Supporting software development as knowledge-intensive and collaborative activity. *Proceedings of the 2006 international workshop on Workshop on interdisciplinary software engineering research,* (WISER), pp. 15–22. DOI: 10.1145/1137 37661.1137666.
25. **Casey, V. (2011)**. Imparting the importance of culture to global software development. *ACM Inroads*, Vol. 1, No. 3, pp. 51–57. DOI: 10.1145/1835428.1835443.
26. **Hernández-López, A., Colomo-Palacios, R., García-Crespo, A., & Soto-Acosta, P. (2010).** Team Software Process in GSD Teams: A Study of New Work Practices and Models. *International Journal of Human Capital and Information Technology Professionals (IJHCITP),* Vol. 1, No. 3, pp. 32–53. DOI: 10.4018/jhcitp.2010070103.
27. **Richardson, I., Casey, V., Mccaffery, F., Burton, J., & Beecham, S. (2012).** A Process Framework for Global Software Engineering Teams. *Information and Software Technology,* Vol. 54, No. 11, pp. 1175–1191, DOI: 10.1016/j.infsof.2012.05.002.
28. **Clerc, V., Lago, P., & Van-Vliet, H. (2011).** Architectural Knowledge Management Practices in Agile Global Software Development. *Proceedings IEEE Sixth International Conference on Global Software Engineering Workshop (ICGSE-W),* pp. 1–8. DOI: 10.1109/ICGSE-W.2011.17.
29. **Bailey, B.P., & Iqbal, S.T. (2008)**. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. (*ACM) Transactions Computer-Human Interaction*, Vol. 14, No. 4, pp. 1–28. DOI: 10.1145/1314683.1314689.
30. **Al-Ani, B., Trainer, E., Ripley, R., Sarma, A., Van Der-Hoek, A., & Redmiles, D. (2008)**. Continuous coordination within the context of cooperative and human aspects of software engineering. *Proceedings international workshop on Cooperative and human aspects of software engineering (CHASE),* pp. 1–4, DOI:10.1145/1370114.1370115.
31. **Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F.J., & Tovar E.**

(2014). Project managers in global software development teams: a study of the effects on productivity and performance. *Software Quality Journal,* Vol. 22, No. 1, pp. 3–19, DOI: 10.1007/s11219-012-9191-x.

32. **Grau, F.G. & Xifra, J.T. (2011).** Zyncro: La Intranet 2.0. *El Profesional de la Información*, Vol. 20, No. 2, pp. 214–218.
33. **Bailey, B.P. & Konstan, J.A. (2006).** On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior*, Vol. 22, No. 4, pp. 685–708, DOI: 10.1016/j.chb.2005.12.009.
34. **Ellis, J. & Kvavilashvili, L. (2000)**. Prospective memory in 2000: Past, present, and future directions. *Applied Cognitive Psychology*, Vol. 14, No.17, pp. 1–9. DOI: 10.1002/acp.767.
35. **Palacio, R.R., Morán, A.L., & González, V.M. (2010).** CWS: An Awareness Tool to Support Starting Collaboration in Global Software Development. *The Open Software Engineering Journal,* Vol. 4, No. 2, pp. 38–51.
36. **Palacio, R.R., Morán, A.L., González, V.M., & Vizcaíno, A. (2012).** Selective availability: Coordinating interaction initiation in distributed software development. *IET Software*, Vol. 6, No. 3, pp. 185–198. DOI: 10.1049/iet-sen.2011.0077.
37. **Chierichetti, F., Epasto, A., Kumar, R., Lattanzi, S., & Mirrokni, V. (2015).** Efficient Algorithms for Public-Private Social Networks. *Proceedings of the 21th (ACM SIGKDD), International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 139–148. DOI: 10.1145/2783258.2783354.
38. **Andrew, M. & Laurie, W. (2011).** Socio-technical developer networks: should we trust our measurements?. *Proceeding of the 33rd International Conference on Software Engineering.,* (ACM). DOI: 10.1145/1985793.1985832.
39. **Poikolainen, T. & Paananen, J. (2007).** Performance Criteria in Inter-Organizational Global Software Development Projects. *Proceedings of the International Conference on Global Software Engineering*. DOI: 10.1109/ICGSE.2007.35.
40. **Verner, J.M., Brereton, O.P., Kitchenham, B.A., Turner, M., & Niazi, M. (2012).** Systematic literature reviews in global software development: A tertiary study. *16th International Conference on Evaluation & Assessment in Software Engineering (EASE),* pp. 2–11. DOI: 10.1049/ic.2012.0001.
41. **Ferrin, D., Bligh, M., & Kohles, J. (2007).** Can I Trust You to Trust Me? A Theory of Trust, Monitoring and Cooperation in Interpersonal and Intergroup Relationships. *Group & Organization Management*, Vol. 32, No. 4, pp. 465–499.
42. **Sarma, A., Van, D., & Hoek, A. (2003).** Palantir: Raising Awareness among Configuration Management Workspaces. *Proceedings of the 25th International Conference on Software Engineering (ICSE)*, pp. 444–454. DOI:10.1109/ICSE.2003.1201222.
43. **Ripley, R.M., Sarma, A., & Van Der-Hoek, A. (2006).** *Using visualizations to analyze workspace activity and discern software project evolution*. University of California, Irvine.
44. **Riemer, K., Scifleet, P., & Reddig, R. (2012).** Powercrowd: Enterprise Social Networking in Professional Service Work: A Case Study of Yammer at Deloitte Australia. *Business and Information Systems*, Vol. 2, pp. 1–18.
45. **Kudos (2014).** Engage Your Employees. http://kudosnow.com/en/main/feature_overview.
46. **Facebook (2014)**. Advertise on Facebook, https://http://www.facebook.com/advertising/.
47. **Mendoza, M., Poblete, B., & Castillo, C. (2010).** Twitter Under Crisis: Can we trust what we RT?. *1st Workshop on Social Media Analytics (SOMA*), pp. 71–79*.* DOI:10.1145/1964858.1964869.
48. **Huang, S.W., Tunkelang, D., & Karahalios, K. (2014).** The Role of Network Distance in LinkedIn People Search, *in The 37th Annual (ACM-SIGIR) Conference.*
49. **IBM (2014).** Social Business, http://www03.ibm.com/press/us/en/presskit/36406.wss.

Article received on 18/11/2016; accepted on 21/08/2017.
Corresponding author is Aurora Vizcaíno.

Research in Computing Science

Series Editorial Board

Research in Computing Science es una publicación trimestral, de circulación internacional, editada por el Centro de Investigación en Computación del IPN, para dar a conocer los avances de investigación científica y desarrollo tecnológico de la comunidad científica internacional. **Volumen 148, No. 5,** mayo de 2019. *Certificado de Reserva de Derechos al Uso Exclusivo del Título* No.: 04-2005-121611550100-102, expedido por el Instituto Nacional de Derecho de Autor. *Certificado de Licitud de Título* No. 12897, *Certificado de licitud de Contenido* No. 10470, expedidos por la Comisión Calificadora de Publicaciones y Revistas Ilustradas. El contenido de los artículos es responsabilidad exclusiva de sus respectivos autores. Queda prohibida la reproducción total o parcial, por cualquier medio, sin el permiso expreso del editor, excepto para uso personal o de estudio haciendo cita explícita en la primera página de cada documento. Distribuida por el Centro de Investigación en Computación, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, México, D.F. Tel. 57 29 60 00, ext. 56571.

Editor responsable: *Grigori Sidorov, RFC SIGR651028L69*

Research in Computing Science is published by the Center for Computing Research of IPN. **Volume 148, No. 5**, May 2019.

Volume 148(5)

Intelligent Learning Environments

María Lucia Barrón Estrada
Ramón Zatarain Cabada
Yasmín Hernández
Carlos A. Reyes García (eds.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2019

ISSN: 1870-4069

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX, DBLP and Periodica

Electronic edition.

ISSN 1870-4069

Table of Contents

Page

ISSN 1870-4069

Synaptix: A Web Platform based on Gamification Techniques for the Study of Clinical Cases

Omar López Chávez, Ignacio N. Márquez, Luis-Felipe Rodríguez, Jorge G. Mendoza León

Sonora Institute of Techonology, Department of Computer and Design, Mexico
omarlopch@gmail.com, ignacio_nmarquez@hotmail.com, luis.rodriguez@itson.edu.mx, jorge.mendoza@itson.edu.mx

Abstract. In this paper we present a Web platform designed to allow medical students and practitioners to study clinical cases on their own. The main objective of this proposal is to address key problems inherent in the traditional study of clinical cases by providing a tool that implements techniques and elements of gamification, simulation, and serious games. The proposed platform offers an improved learning experience through a virtual environment that provides an alternative method for training and interpretation of clinical cases for medical examinations. Medical students and practitioners can play the role of a real doctor in a simulated office. In particular, the platform allows medical students and practitioners to learn through their mistakes without hurting human beings. In addition, this platform is designed to allow users to add new clinical cases and make them available for study. The platform was validated in a local hospital by 8 medical practitioners. The participants indicated that the platform design, the tutorials included, and the ease of use factors are satisfactory.

Keywords: medical student, clinical case, gamification, simulation, serious game.

1 Introduction

Gamification is the process of changing a set of traditional actions to an attractive gaming experience for the user [13]. Matallaoui *et al.* [11] define gamification as "the use of game design elements in a context not related to the game, is an interdisciplinary tool, where users are motivated to achieve certain behavioral or psychological results". Gamification also enables the development of immersive games in virtual environments in which users are encouraged to perform desired actions. In the academic field, gamification serves as a tool to facilitate teaching and learning processes through collaborative environments [2].

Serious games (SGs) have a high impact as an instructional tool that benefits from traditional game concepts and information and communication technologies. Serious games have allowed the implementation of simulations and realistic

virtual environments, where players can experience adventures while acquiring, practicing, and verifying knowledge. This represents a significant opportunity for 21st century educators and trainers to improve their educational tools [1].

The traditional method for studying clinical cases is through information sources such as books, articles, and automated tests. However, this method usually leads medical students to a state of saturation, stress, and anxiety given that, for example, the quantity and complexity of clinical cases to study in exam periods is high. Moreover, the feedback received from this type of information sources is limited. The study of clinical cases by medical students and practitioners also takes advantage of the monitoring of patients in hospitals. This study of clinical cases requires observation and analysis for long periods of patients with diverse conditions, diseases, signs, and symptoms [9]. However, in hospitals such as those known as "third level hospitals", only critical patients have long stays. The majority of patients are hospitalized for short periods of time, which makes it difficult the generation and access of medical students to a greater amount of knowledge that improves their learning experience and the acquisition of skills related to clinical cases that depend on hospitalized patients.

Lifshitz [8] indicates that clinical cases can not be learned through memorization or readings or through distance education strategies. In fact, the analysis of clinical cases has a very strong affective component because it implies confrontation with illness and suffering. The discussion about the limits of the teaching of clinical cases has not been solved, but this type of learning usually requires abilities for communication, physical examination, treatment, and clinical reasoning. Lifshitz [8] also proposes a well defined structure to organize and study clinical cases: 1) the approach to the patient, 2) the collection of information, 3) the analysis of the information collected, 4) clinical procedures, 5) the diagnostic decision, 6) the decision therapist, and 7) the decision Prognosis. In this context, although there is great interest in the design and application of guidelines in clinical practices for the prevention and care of diverse health situations, greater attention must be paid to its implementation and effectiveness in various practical scenarios [5].

The simulation of clinical cases involves a set of techniques that facilitate medical students and practitioners the acquisition of knowledge and skills. In particular, techniques and methods from fields such as artificial intelligence, virtual and augmented reality, and human-computer interaction have enabled the development of platforms that incorporate virtual scenarios, simulation models, and multimedia materials to simulate different real situations (e.g., for the analysis of clinical cases) [4]. Although simulation platforms do not replace the real scenarios, these allow students to learn and practice in controlled media, contributing to improve their skills and decrease the anxiety when performing an exam or procedure. This type of platform also accelerates learning and enriches the true interactions with the patients, helping medical students to avoid states of saturation experienced with traditional learning methods [9].

In this paper, we present a platform designed to allow medical students and practitioners to study clinical cases on their own. This platform represents an

attempt to address key problems inherent in the traditional study of clinical cases by providing a tool that implements techniques and elements of gamification, simulation, and serious games. Its design takes into account the elements and strategies that according to medical students and practitioners are required to learn and practice clinical cases [10]. The paper is structured as follows. In Section 2 we discuss related work. The proposed platform and corresponding validation are presented in Section 3 and Section 4, respectively. Finally, concluding remarks and future work are discussed in Section 5.

2 Related Work

The website *The New England Journal of Medicine* presents interactive medical clinic cases designed according to the following interaction process: presentation of the case, medical history of the patient, information of the physical examination, and finally, performs a test to provide feedback and solutions of correct and incorrect answers [6]. However, although this website shows to users the percentage of the result and the studies carried out, key elements of gamification and serious games are not considered such as *dashboard*, *unlock*, and *challenge*. Moreover, the interaction is based on text and 2D graphics, leaving aside the implementation of 3D scenarios and simulations. The inclusion of additional clinical cases by users is not allowed.

Nevin *et al.* [12] developed the Kaizen-Internal Medicine (Kaizen-IM) software that includes elements of gamification. This tool involved a large number of residents in a medical contest that facilitated the acquisition of new knowledge in the academic period 2012-2013 in two training programs IM (internal medicine) in the USA: the residency program in Internal Medicine at the University of Alabama at Birmingham (UAB) and the University of Alabama Program at Huntsville (UAH). The data was recorded at participant level and question. The analyzes focused on the acceptance, use, and determination of the factors associated with the loss of players (attrition) and the retention of knowledge. The Kaizen-IM data provided information on modifiable factors associated with student attrition and retention of knowledge that can serve to further enhance the educational benefits of this strategy for students. This tool is an attempt to demonstrate the benefits of incorporating elements of gamification in the learning process of medical students. However, this software does not implement 3D simulations, but is based on strategies based on questions and answers.

Leba *et al.* [7] proposed an application for the training of medical students in the field of anatomy with computed tomography images, using elements of gamification, simulation, and serious games. It was designed in the context of an educational software. This proposal represents an attempt to support modern and practical methods of examination based on real cases useful for medical students and teachers. However, this application for training was not validated in a case study, the authors proposed only design guidelines.

3 Synaptix

The Sypnatix platform was designed for the study of clinical cases that follow the clinical practice guide of the federal government of Mexico [3]. Synaptix implements elements and techniques of gamification, simulation, and serious games. In particular, its design is based on a previous study by Marquez *et al.* [10] in which data was collected from medical students and practitioners in order to 1) identify key elements and learning strategies for the study and practice of clinical cases, and 2) define how these elements and strategies should be taken into account in the design of a learning platform that incorporates elements of gamification for the study of clinical cases (see Figure 1) [10].

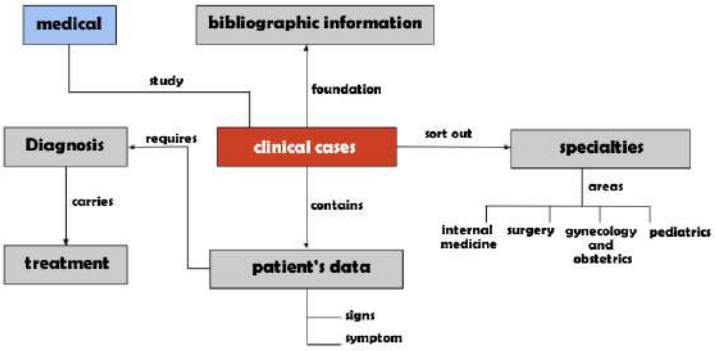


Fig. 1. Elements involved in the study of clinical cases [10].

The Synaptix platform incorporates the following elements of gamification (which are common elements reported in the literature [14]):

- Points: represent the way in which the player is observed, classified, and guided. In Synaptix, the user starts with 0 points. The points are cumulative and depend on the correct answers provided by the user.
- Badges: mark the fulfillment of goals and the constant progress of the game. Badges are activated as the user completes clinical cases on Synaptix.
- Levels: a marker for players to know where they are in a gaming experience. In Synaptix, levels are activated as progress is made in solving clinical cases.
- Dashboard: an ordered list of names and its corresponding score. Synaptix makes available information about the achievements of each user.
- Unlock: allows players to access another achievement after certain requirements are met. In Synaptix, objectives are unlocked as the user resolves clinical cases.
- Challenges: offer players an address so they know what to do within the world of the gamified experience. In Synaptix, the challenges are associated with obtaining badges.

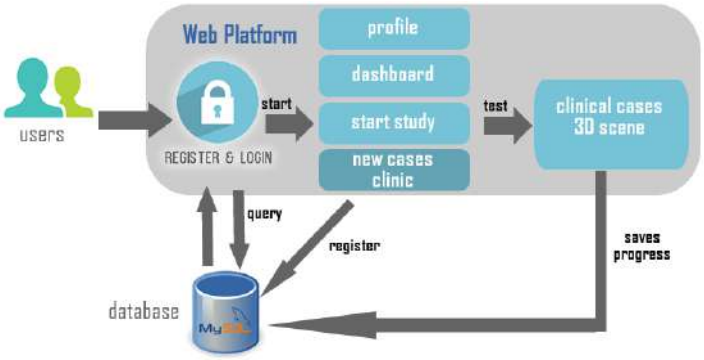


Fig. 2. The architecture of Synaptix.

Figure 2 shows the main components of the Sypnatix's architecture and their relationship: 1) the elements of gamification mentioned above, 2) a 3D virtual scenario, and 3) components related to the data management. These components create an interactive experience by enabling medical students to acquire, practice, and verify knowledge, which are key elements of SG and simulation.

Fig. 3. Simulation of a doctor office and a virtual patient.

The virtual scenario provided by Synaptix for the study and practice clinical cases is shown in Figure 3. This virtual scenario simulates a medical office and a virtual patient. Medical students interact with the virtual patient by physically examining, for example, its lungs and head. Additional information related to clinical records or results of clinical studies (data that is sometimes taken into account in clinical cases) is also displayed in the scenario. After the medical student analyzed the data provided in the clinical case (i.e., physically examined the virtual patient and its medical records), a series of questions and possible

answers are displayed to evaluate whether the diagnosis and treatment suggested by the student are correct. Synaptix provides feedback to medical students once an answer is submitted. If the answer is incorrect, feedback about medications or treatments is provided. The points obtained and the assigned badges are activated in the user's profile as shown in Figure 4.

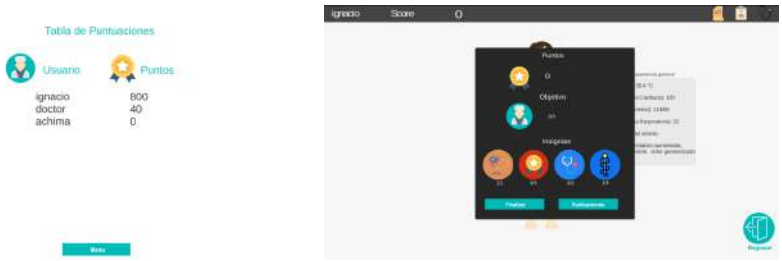


Fig. 4. Dashboards that show the scores and points obtained by medical students.

Although Synaptix is designed for the study of clinical cases related to the areas of internal medicine, surgery, gynecology and obstetrics, and pediatrics, currently only clinical cases of internal medicines are included. Nevertheless, Synaptix allows medical practitioners to include new clinical cases which are then available for their analysis by medical students. The tools used in the development of the Synaptix Web platform were Unity game development platform, Php scripting language, and the Mysql database server. In order to carry out the tests, it was posted on a web server: http://arevolution.com.mx/synaptix/

4 Validation

Synaptix was validated in a private local hospital by 8 medical students and practitioners (4 female and 4 male). The validation session consisted of an introduction by the authors about the functionality and characteristics of the platform. Afterwards, the participants used the platform to practice available clinical cases and answered a questionnaire. The instrument consisted of 29 items to measure 1) the design of the platform, 2) the instructions provided by Synaptix, and 3) easy of use factor. Figure 5 and Figure 6 show the results obtained in this evaluation phase.

The comments provided by participants include the following: 1) provide further feedback or suggest additional information sources to the user once a clinical case is carried out, 2) allow the user to make more questions to the virtual patient, 3) indicate the specific areas the medical student needs to reinforce in order to achieve better results, 4) provide greater details in the diagnosis, 5) include better medical images, and 6) include more references about the clinical cases presented. Finally, some participants emphasized the importance of badges and individual scores as incentives.

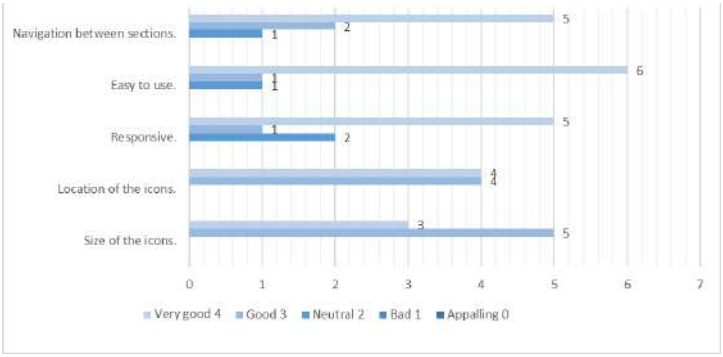


Fig. 5. Results of evaluating the design of Synaptix.

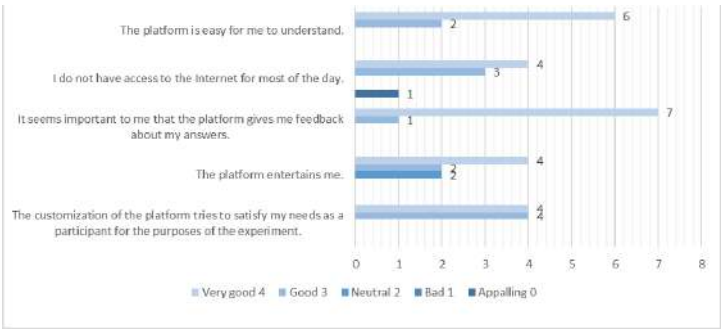


Fig. 6. Results of evaluating the easy of use of Synaptix.

5 Conclusions and Future Work

The main contribution of this paper is the design and implementation of a platform that 1) incorporates elements of gamification, serious games, and simulation, and 2) takes into account the elements and strategies that according to medical students and practitioners are required to learn and practice clinical cases. The Synaptix Web platform is a gamified tool that attempts to serve as an alternative method for the study of clinical cases, avoiding states of saturation, fatigue, and anxiety in medical students and practitioners. Synaptix was validated in a private local hospital by 8 medical students and practitioners. The results demonstrated that participants find that the factors associated with the *design* and *easy of use* of the platform is in general satisfactory. Future research involves an evaluation to measure complex individual's aspects such as motivation, learning, and user engagement as well as the incorporation of virtual

reality and augmented reality components in order to create more immersive scenarios. Furthermore, AI techniques will be incorporated to enable the virtual patients to develop some human-like behaviors.

Acknowledgment. This work was supported by PFCE 2018.

References

1. Arnab, S., Lim, T., Carvalho, M.B., Bellotti, F., De Freitas, S., Louchart, S., Suttie, N., Berta, R., De Gloria, A.: Mapping learning and game mechanics for serious games analysis. British Journal of Educational Technology 46(2), 391–411 (2015)
2. Burke, B.: Gamify: How gamification motivates people to do extraordinary things. Routledge (2016)
3. Cenetec: Clinical practice guide. `https://cenetec-difusion.com/gpc-sns/` (2018)
4. Chen, L., Day, T.W., Tang, W., John, N.W.: Recent developments and future challenges in medical mixed reality. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 123–135. IEEE (2017)
5. DiCenso, A., Guyatt, G., Ciliska, D.: Evidence-Based Nursing-E-Book: A Guide to Clinical Practice. Elsevier Health Sciences (2014)
6. Group, N.: interactive medical case. `https://www.nejm.org/multimedia/interactive-medical-case` (2018)
7. Leba, M., Ionica, A., Apostu, D.: Educational software based on gamification techniques for medical students. In: Proceedings of the 5th International Conference on Applied Informatics and Computer Theory (AICT). pp. 225–230 (2014)
8. Lifshitz, A.: La nueva clínica. Intersistemas Editores (2014)
9. Makransky, G., Bonde, M.T., Wulff, J.S., Wandall, J., Hood, M., Creed, P.A., Bache, I., Silahtaroglu, A., Nørremølle, A.: Simulation based virtual learning environment in medical genetics counseling: an example of bridging the gap between theory and practice in medical education. BMC medical education 16(1), 98–107 (2016)
10. Márquez, I., Mendoza, J.G., Rodríguez, L.F.: Identificación de elementos clave en el estudio de casos clínicos para su gamificación. In: Prieto, M., Pech, S., Francesa, A. (eds.) Tecnologías y Aprendizaje. Investigación y Práctica, pp. 132–139 (2018)
11. Matallaoui, A., Herzig, P., Zarnekow, R.: Model-driven serious game development integration of the gamification modeling language gaml with unity. In: 48th Hawaii International Conference on System Sciences (HICSS). pp. 643–651. IEEE (2015)
12. Nevin, C.R., Westfall, A.O., Rodriguez, J.M., Dempsey, D.M., Cherrington, A., Roy, B., Patel, M., Willig, J.H.: Gamification as a tool for enhancing graduate medical education. Postgraduate medical journal pp. 121–130 (2014)
13. Robson, K., Plangger, K., Kietzmann, J.H., McCarthy, I., Pitt, L.: Is it all a game? understanding the principles of gamification. Business Horizons 58(4), 411–420 (2015)
14. Vargas-Enríquez, J., García-Mundo, L., Genero, M., Piattini, M.: Análisis de uso de la gamificación en la enseñanza de la informática. In: Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática. pp. 105–112. Universitat Oberta La Salle (2015)

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

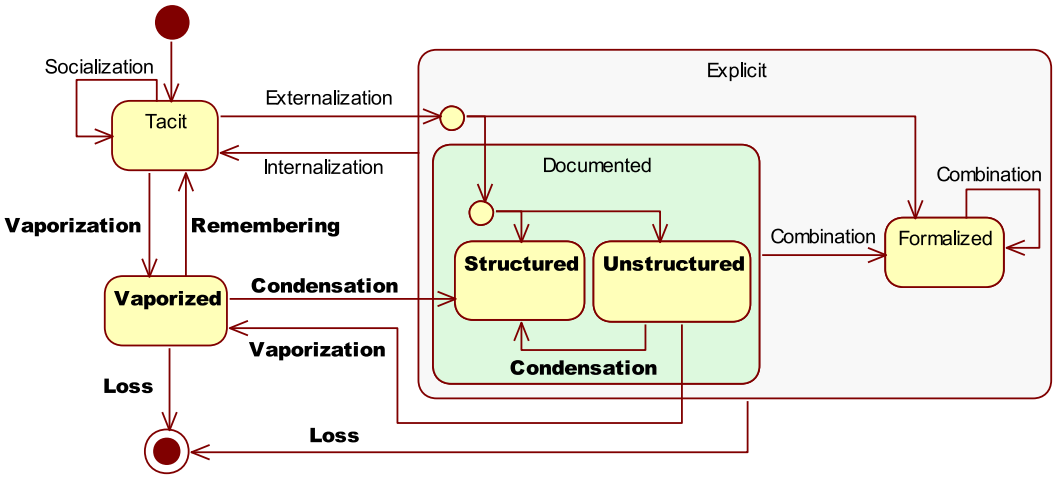


Fig. 1. UML state diagram representing SECI model with documented and formalized sub-states, and extended with the vaporization and condensation concepts. States and transitions in bold are product of our SECI model extension.

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/.

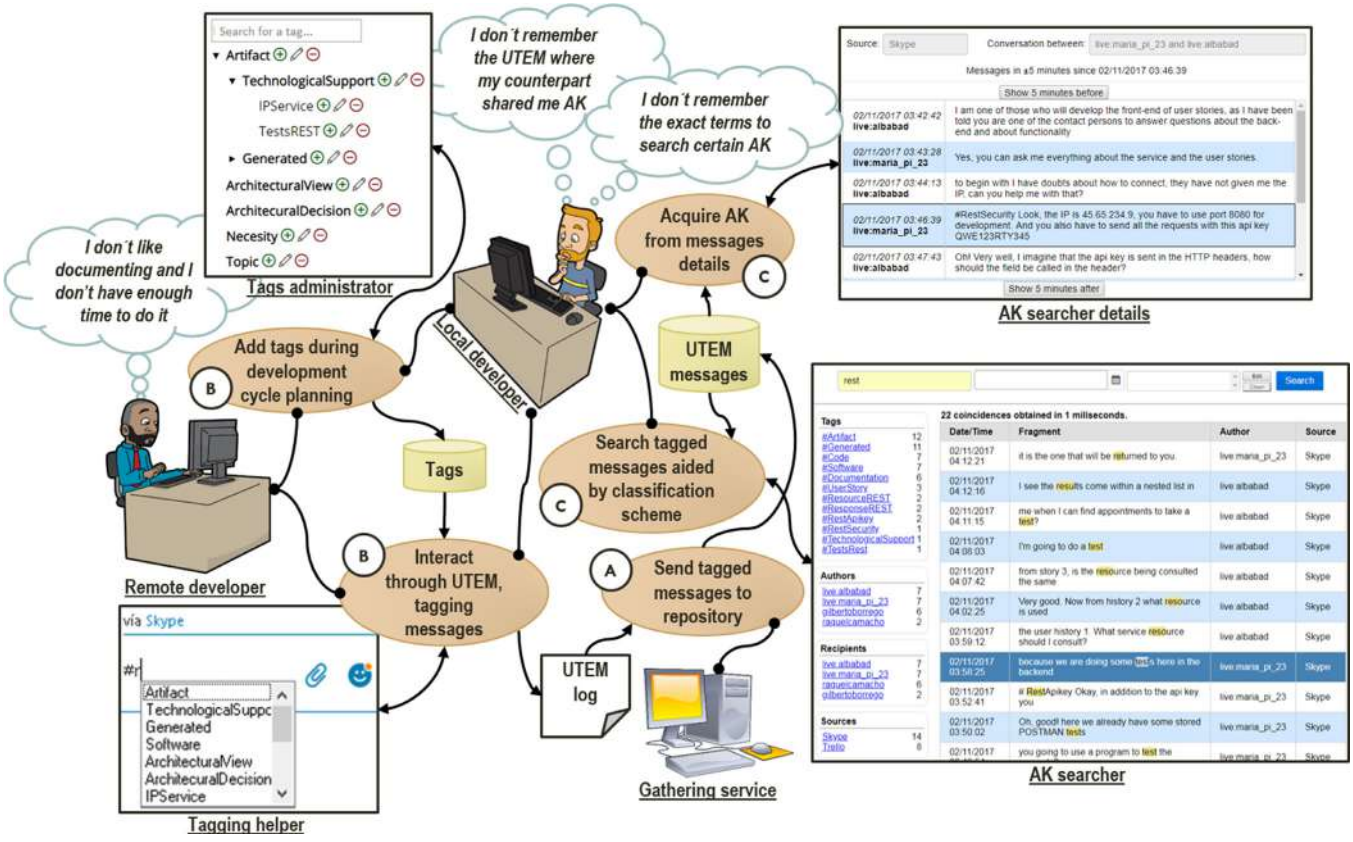


Fig. 2. ArchiKCo rich picture with activities (ovals) corresponding to the three elements of AK Condensation concept, A = Accessible UTEM logs information, B = UTEM logs Classification mechanism, C = AK searching mechanism. Bulleted lines represent links between activities and who performs them. Arrowed 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).

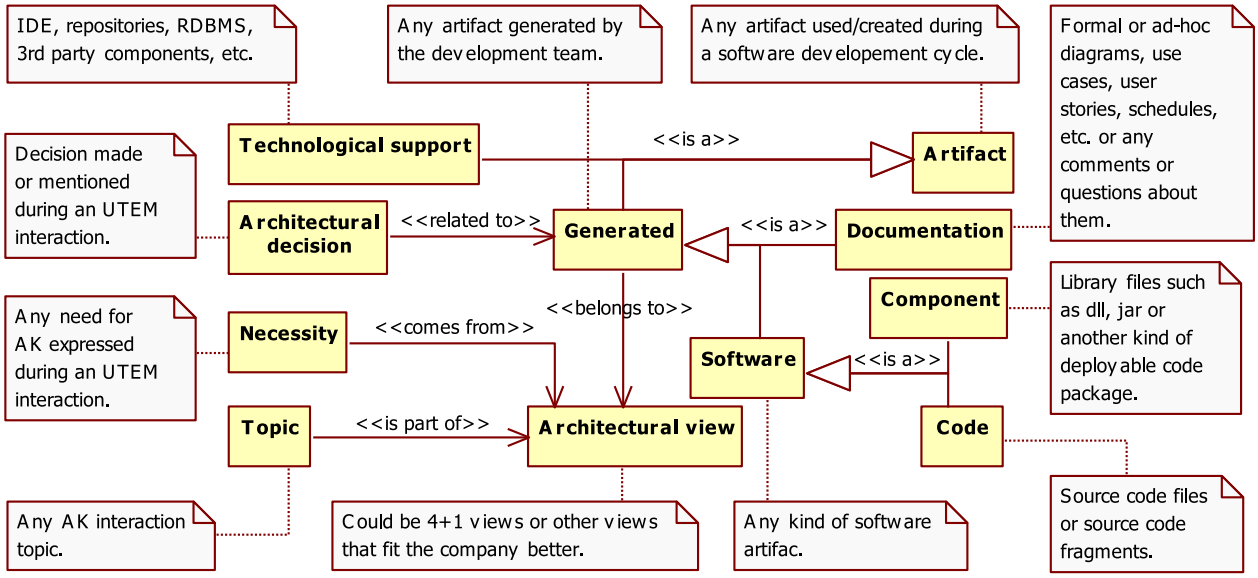


Fig. 3. Conceptual model based on UML, representing the aspects involved in AK articulation through use of UTEM by AGSD teams (reproduced from [19]).

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

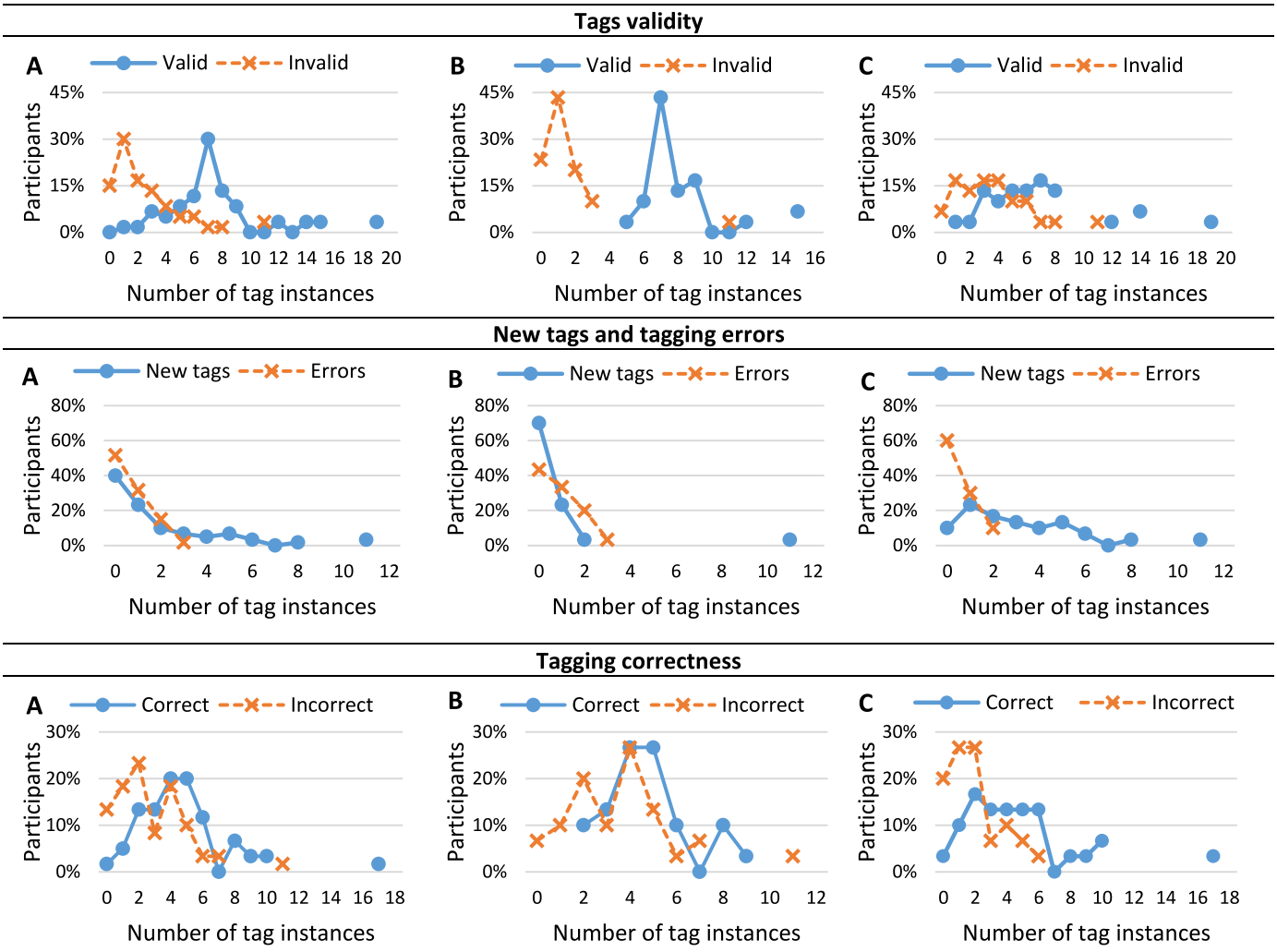


Fig. 4. Tagging behavior expressed by tag instances per participant.A = Mexican developers and UCLM students, B = Mexican developers only, C = UCLM students only. Outliers correspond to identified participants (< 2) who used many valid/invalid tag instances, or many new tags instances, or many correct/incorrect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

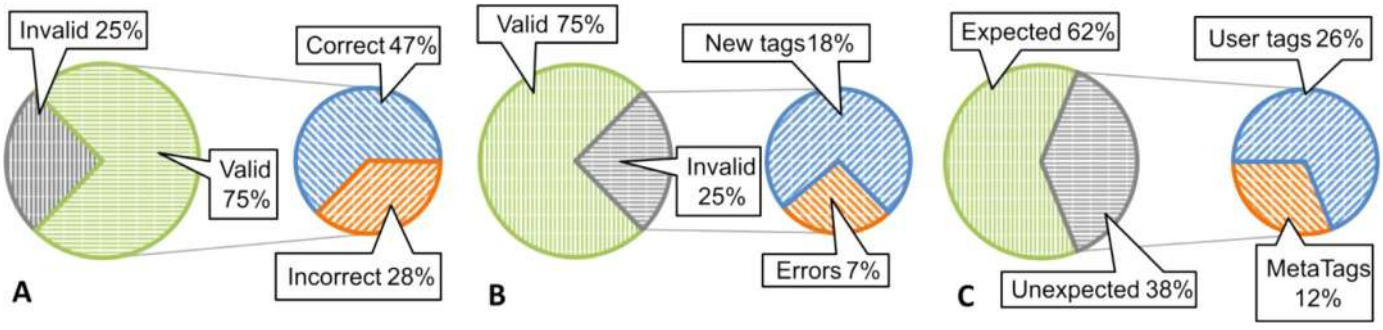


Fig. 5. A = Distribution of all tag instances (valid and invalid), detailing valid tags. B = Distribution of all tag instances (valid and invalid), detailing invalid tags. C = Distribution of correct tag instances (expected and unexpected), detailing unexpected.

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

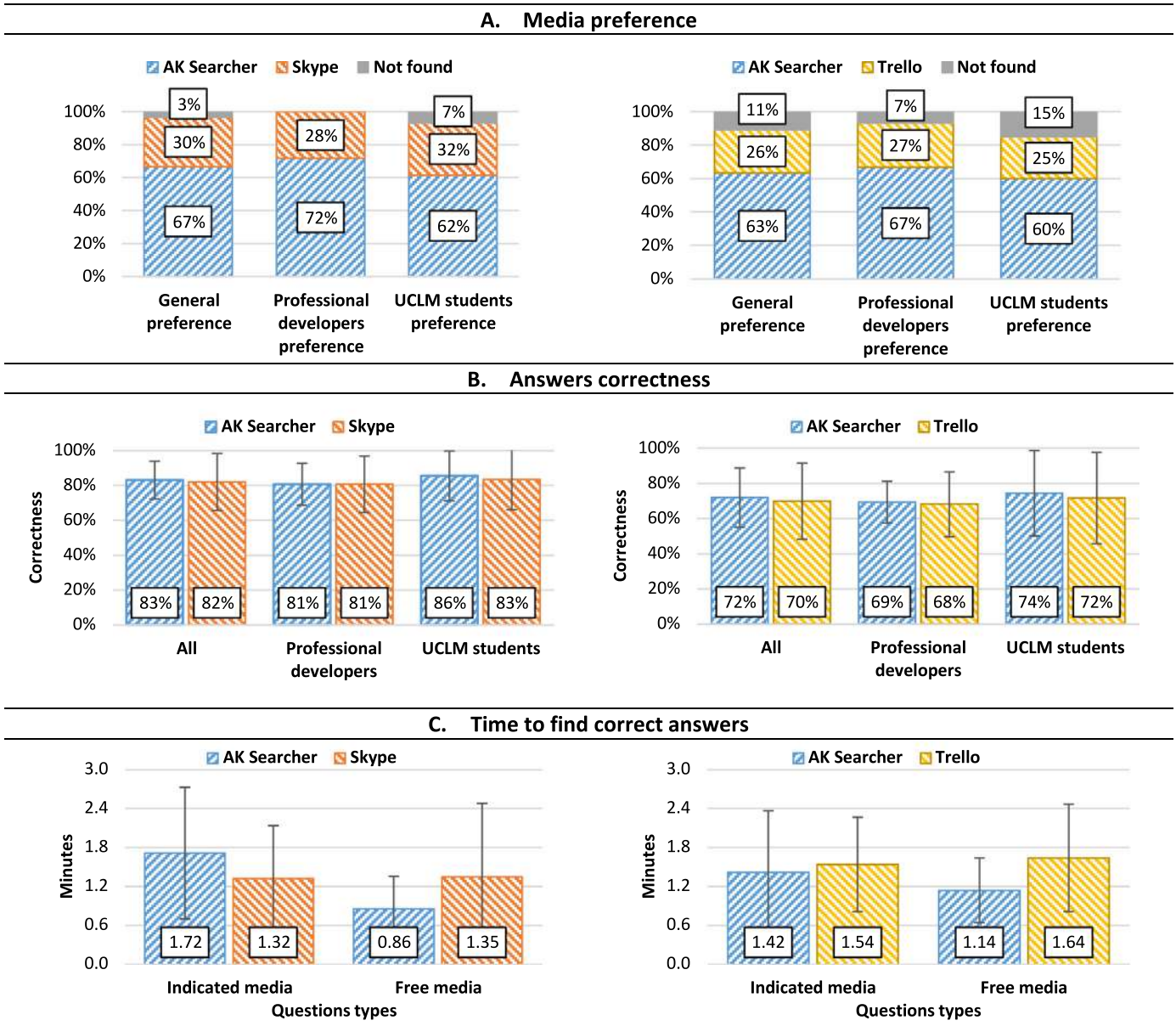


Fig. 6. A = Media preference of the participants as regards questions in which they had free choice. B = Comparison between correctness of answeres obtained using AK Searcher and Skype. C= Answering average times per question type. Vertical lines in B and C represent standard deviation.

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.