Information and Software Technology 54 (2012) 1215–1228

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

Analyzing the understandability of Requirements Engineering languages for CSCW systems: A family of experiments

Miguel A. Teruel [a], Elena Navarro [a,*], Víctor López-Jaquero [a], Francisco Montero [a], Javier Jaen [b], Pascual González [a]

[a] LoUISE Research Group, Computing Systems Department, University of Castilla—La Mancha, Avda. España s/n, 02071 Albacete, Spain

[b] ISSI, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n. 46022 Valencia, Spain

ARTICLE INFO

Article history:
Received 30 December 2011
Received in revised form 1 June 2012
Accepted 3 June 2012
Available online 12 June 2012

Keywords:
Collaborative systems
Awareness
Controlled experiment
Understandability
Requirements Engineering
Meta-analysis

ABSTRACT

Context: A collaborative system is a special kind of software whose users can perform collaboration, communication and collaboration tasks. These systems usually have a high number of non-functional requirements, resulting from the users' need of being aware of other users with whom to collaborate, that is, the workspace awareness.

Objective: This paper aims at evaluating two Requirements Engineering languages i^* and CSRML (an extension of i^*) in order to determine which is the most suitable one to specify requirements of collaborative systems, taking into account their special characteristics regarding collaboration and awareness.

Method: We performed a family of experiments comprising an original experiment and two replicas. They were performed by 30, 45 and 9 Computer Science students, respectively, from Spain and Argentina. These subjects filled in two understandability questionnaires once they analyzed the requirements models of two systems: an e-learning collaborative system and a conference review system with some collaborative aspects support. Both models were specified by using the evaluated languages.

Results: The statistical analysis of the family of experiments showed that the understandability was higher for the models specified with CSRML than for those specified with i^*, especially for collaborative aspects. This result was also confirmed by the meta-analysis conducted.

Conclusions: CSRML surpasses i^* when modeling collaborative systems requirements models due to the specific expressiveness introduced to represent collaboration between users and awareness and the new resorts to manage actors and roles.

1. Introduction

Internet is continuously evolving not only in the way it is developed, but also in the way it is browsed and exploited. Internet sites have evolved from simple static sites created by a single developer, which just enabled visitors to read their contents, to complex dynamic sites, which can be created and modified by users in a collaborative way. It is worth taking a look at the top 1000 most visited pages in 2011 [1] and see that, except for search engines and news websites, almost all of the top 100 ones are collaborative webs. In addition, the just mentioned search engines and news web sites are becoming also collaborative by including new social features, such as the possibility of adding comments or recommendations to news or search results.

Nowadays, even classic applications like text processors are collaborative. For instance, Google Docs [2] enables several users to edit a text document simultaneously. These collaborative text processors are a good example of CSCW (*Computer Supported Cooperative Work*) systems [3], which are systems whose users can perform collaboration, communication and coordination tasks (*3C*). Collaborative systems, in a similar way to classical single-user systems, have to be specified by means of a set of requirements, whose accuracy and suitability are key to achieve the quality of the developed system. The main difference between the requirements of single-user systems and CSCW systems is the highly non-functional nature of the latter, because of the users' need of being aware of the presence of other users with whom to perform the above mentioned 3C tasks, that is, the *Workspace Awareness* (WA).

WA was defined by Gutwin as "*the up-to-the-moment understanding of another person's interaction within a shared workspace*" [4]. This WA involves knowledge about *where* others are working, *what* they are doing *now*, *when* a user performed some action, and *what* they are going to do *next*. Tables 1 and 2 show the WA elements related to the present and to the past, respectively.

* Corresponding author. Tel.: +34 967 599 200x2624; fax: +34 967 599 343.
E-mail addresses: miguel@dsi.uclm.es (M.A. Teruel), enavarro@dsi.uclm.es (E. Navarro), victor@dsi.uclm.es (V. López-Jaquero), fmontero@dsi.uclm.es (F. Montero), fjaen@dsic.upv.es (J. Jaen), pgonzalez@dsi.uclm.es (P. González).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.06.001

Table 1
Elements of WA related to the present.

Category	Element	Specific questions
Who	Presence	Is anyone in the workspace?
	Identity	Who is participating? Who is that?
	Authorship	Who is doing that?
What	Action	What are they doing?
	Intention	What goal is that action part of?
	Artifact	What object are they working on?
Where	Location	Where are they working?
	Gaze	Where are they looking?
	View	Where can they see?
	Reach	Where can they reach?

Table 2
Elements of WA related to the past.

Category	Element	Specific questions
How	Action history	How did that operation happen?
	Artifact history	How did this artifact come to be in this state?
When	Event history	When did that event happen?
Who	Presence history	Who was here, and when?
Where	Location history	Where has a person been?
What	Action history	What has a person been doing?

In order to deal with the specification of these special type of systems, we conducted an initial test, by using DESMET [5], to check which is the most adequate Requirements Engineering technique to model both awareness and quality requirements of CSCW systems [6]. The analyzed techniques were Use Cases [7,8], Goal-Oriented [9] and Viewpoints [10], and we concluded that Goal-Oriented techniques were the most promising ones for specifying this kind of requirements. Next, we conducted a second study comparing the suitability of three Goal-Oriented approaches for modeling CSCW systems requirements [11]. The compared proposals were NFR Framework [12], i^* Framework [13] and KAOS Methodology [9]. The analysis of the results showed that i^* was the most suitable proposal.

Despite the previous studies that determined that i^* was the most promising technique, we identified several issues when modeling collaborative systems with this language, such as the lack of expressiveness for awareness requirements, imprecise 3C task representation, and a not fully suitable management of roles and actors. Due to these problems, the original i^* language was extended to try to solve the above mentioned deficiencies by creating CSRML (Collaborative Systems Requirements Modeling Language) [14].

As Basili et al. [15] claim, families of experiments can be used to draw relevant conclusions that an experiment alone cannot provide. For this reason, in this paper, a family of experiments is presented whose general goal is to test which language, CSRML or i^*, has a better understandability when modeling requirements of CSCW systems. This family of experiments has been conducted by following the Wholin et al.'s guidelines [16] and it has consisted of an experiment and two replicas, all of them carried out by students of Computer Science from three different universities. These students were asked to fill in two understandability questionnaires once they had analyzed the requirements models of an e-learning collaborative activity and a conference review system with collaborative aspects support. Both requirements models were specified with the tested languages.

As aforementioned, *understandability* was the criteria analyzed in the family of experiments. Understandability has been defined by the International Organization for Standardization (ISO) as "the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use" [17]. Indeed, considering the requirements specification as one of the products of the software process, it becomes critical to provide a specification understandable by every stakeholder to develop a suitable software. This fact is also supported by the IEEE830-1998 standard for requirements specifications [18], since it considers an understandable specification as one of the means to decrease the development effort by reducing later redesign, recoding, and retesting. Moreover, understandability has also been considered as a quality attribute in several quality standards. For instance, it is one of the five quality characteristics that an engineering model must have along with *abstraction*, *accuracy*, *predictiveness* and *inexpensiveness* [19]. Furthermore, ISO includes understandability as one of the six sub-characteristics of the usability quality factor. More precisely, usability is defined as "the capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions" [17].

Based on these considerations, this paper is structured as follows: Section 2 presents several works related to CSCW and Empirical Software Engineering, focusing on the evaluation of Requirements Engineering models. Next, in Section 3, i^* and CSRML are briefly introduced. In Section 4 the family of experiments is presented by describing how it has been carried out and its main results. Finally, some conclusions and future work are drawn in Section 5.

2. Related works

In traditional software applications, only one user is able to interact with the application. Therefore, there is no support for collaboration among users. Nevertheless, in the late eighties, Computer Supported Cooperative Work (CSCW [3,20]) emerged as a new discipline to study collaborative systems, a special kind of software in which users can collaborate with each other by sharing resources, chatting, etc.

One of the key elements for the construction of CSCW systems is the proper application of a methodology that not only guides their development, but also takes into account the peculiarities that make them different from other systems. An example is AMENITIES [21], a methodology presented by Garrido, for the development of cooperative systems based on behavior models and tasks. This methodology pursues the development of a system model from a requirements model by following a straightforward method that allows stakeholders to represent tasks and relate them to both the users responsible for their accomplishment (by means of roles) and other relevant concepts of the problem's domain (artifacts, information, etc.). Despite being a complete development methodology, AMENITIES has an important lack of expressiveness for the representation of the *Workspace Awareness* (WA), a cornerstone for the development of this kind of systems, as aforementioned in Section 1. This is due to the use of a classical RE technique, Use Cases, that makes it difficult to represent non-functional requirements related to the WA [6].

Based on the aforementioned methodology, Molina et al. presented CIAM [22], a methodological proposal for the development of user interfaces for CSCW systems. This methodology improves AMENITIES by combining interaction, collaboration and information sharing aspects and supporting WA in a better way. Nevertheless, it also uses the same Requirements Engineering (RE) technique than AMENITIES. Therefore, CSCW requirements cannot be properly specified. Also based on Garrido's proposal, Penichet et al. presented TOUCHE [23], a process model for the development of user interfaces for CSCW systems, which is user-centered and driven by tasks. This methodology does improve the requirements specification by enriching the Use Case specification with information