UNIVERSIDAD DE CASTILLA-LA MANCHA

University of Castilla - La Mancha

An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems

Technical Report # DIAB-11-01-1

Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero, Pascual González

January 2011

A collaborative system is a distributed software which allows several users to work together and carry out collaboration, communication and coordination tasks. To perform these tasks, the users have to be aware of other user's actions, usually by means of a set of awareness techniques. When we are defining a collaborative system, the awareness techniques can be considered as non-functional requirements bounded to some quality factors, such as usability. However, serious flaws can be found during the specification of these systems if we use the usual Requirement Engineering techniques available, because their expressiveness limitations when dealing with non-functional requirements. In this report an empirical evaluation is introduced to determine if these techniques are really appropriate to model groupware requirements and which is the best approach to specify this kind of systems. With this aim, a collaborative text editor is used to evaluate whether the current techniques for Requirement Engineering are appropriated or not, exploiting the relation between awareness capabilities and standard quality factors.

An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems

Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero, Pascual González
LoUISE Research Group
Computing Systems Department
University of Castilla - La Mancha
{MiguelAngel.Teruel, Elena.Navarro, VictorManuel.Lopez, Francisco.MSimarro, Pascual.Gonzalez}@uclm.es

Abstract— ***A collaborative system is a distributed software which allows several users to work together and carry out collaboration, communication and coordination tasks. To perform these tasks, the users have to be aware of other user's actions, usually by means of a set of awareness techniques. When we are defining a collaborative system, the awareness techniques can be considered as non-functional requirements bounded to some quality factors, such as usability. However, serious flaws can be found during the specification of these systems if we use the usual Requirement Engineering techniques available, because their expressiveness limitations when dealing with non-functional requirements. In this paper an empirical evaluation is introduced to determine if these techniques are really appropriate to model groupware requirements and which is the best approach to specify this kind of systems. With this aim, a collaborative text editor is used to evaluate whether the current techniques for Requirement Engineering are appropriated or not, exploiting the relation between awareness capabilities and standard quality factors.***

I. Introduction

At the end of the eighties, Computer Supported Cooperative Work (CSCW) emerged as a new area of research that focuses on the study of the human behavior in the working context and also on the design of tools (groupware) that support workgroups [*1*]. The problem of this kind of systems is to maintain so called the workspace awareness.

Workspace Awareness (WA) is the up-to-the-moment understanding of another person's interaction within a shared workspace. Workspace awareness involves knowledge about *where* others are working, *what* they are doing *now*, and *what* they are going to do *next* [*2*].

In a face-to-face workspace, awareness of someone else is relatively easy to maintain, and the mechanisms of collaboration are natural, spontaneous, and unforced. Unfortunately, workspace awareness is much harder to maintain in groupware workspaces than in face-to-face environments, and it is often difficult or impossible to determine who else is in the workspace, where they are working, and what they are doing.

Gutwin [*2*] presents a conceptual framework to establish what information makes up workspace awareness. The basic the elements is the set of questions "who, what, where, when, and how". That is, when we work with others in a physical shared space, we know who we are working with, what they are doing, where they are working, when various events happen, and how those events occur.

TABLE 1: ELEMENTS OF WORKSPACE AWARENESS 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 WORKSPACE AWARENESS RELATING 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?

Tables 1 and 2 show these elements and list the questions that each element can answer. Table 1 contains those elements that relate to the present, and Table 2 contains those that relate to the past. The elements are all commonsense things that deal with interactions between a person and the environment.

In this context a proper specification of the system, identifying clearly the requirements of the system-to-be, specially the awareness requirements, is one of the first steps to overcome this problem. The awareness requirements can be considered non-functional requirements (NFR) or extra-functional requirements (EFR), because they are usually constraints regarding quality (e.g. functionality, usability) [*3*]. However, the specification of this kind of requirements is not a trivial issue because of the high number and diversity of requirements they are related to, and their high impact in terms of the final architecture of the system. Therefore, the proper selection of the requirement specification technique becomes a challenging and important decision.

In this paper, we study the applicability of three Requirement Engineering (RE) techniques (Use Cases [*4*], Viewpoints [*5*], and Goal-Oriented[*6*]) for the specification of collaborative systems, paying special attention to the awareness requirements. In order to carry out this study, we have specified some awareness requirements of a real system (Google Docs [*7*]). Once the system is modeled, an empirical analysis has been done in order to compare these different techniques.

This paper is structured as follows. After this introduction, in section II, we analyze three RE techniques applicable to awareness requirements for CSCW systems. In section III, we present an example of a widely known collaborative system: Google Docs. In section IV, an empirical evaluation of the use of the previous techniques for modeling awareness requirements in Google Docs is presented. In section V, we propose our conclusions and future works.

II. RE TECHNIQUES FOR COLLABORATIVE SYSTEMS

In order to enhance the legibility of this work, in the following sections we describe briefly the main concepts underlying the three analyzed RE techniques, namely: Goal-Oriented, Use Cases and Viewpoints.

A. Goal-Oriented requirement specification

In the context of Requirements Engineering, the Goal-Oriented Requirements Engineering approach [*8*] has proven to be useful in eliciting and defining requirements. More traditional systems analysis techniques, such as Use Cases, only focus on establishing the features (i.e. activities and entities) that a system will support. Nevertheless, Goal-oriented proposals focus on why systems are being constructed by providing the motivation and rationale to justify the Software Requirements. They are not only useful for analyzing goals, but also for elaborating and refining them.

A Goal Model is built as a directed graph by means of a refinement of the systems goals (see Figure 5). This refinement lasts until goals have enough granularity and detail so as to be assigned to an agent (software or environment) so that they are verifiable within the system-to-be. This refinement process is performed by using AND/OR/XOR refinement relationships.

There are a wide number of proposals ranging from elicitation to validation activities in the RE process (see [*9*] for an exhaustive survey). However, some concepts are common to all of them:

- *Goal* describes why a system is being developed, or has been developed, from the point of view of the business, organization or the system itself. In order to specify it, both functional goals, i.e., expected services of the system, and *softgoals* related to the quality of service, constraints on the design, etc should be determined.
- *Agent* is any active component, either from the system itself or from the environment, whose cooperation is needed to define the operationalization of a goal, that is, how the goal is going to be provided by the system-to-be. This operationalization of the goals is exploited to maintain the traceability throughout the process of software development.
- *Refinement Relationships*: AND/OR/XOR relationships allow the construction of the goal model as a directed graph. These relationships are applied by means of a refinement process (from generic goals towards sub-goals) until they have enough granularity to be assigned to a specific operationalization.

It must be pointed out that one of the main advantages exhibited by this approach is that it introduces mechanisms for reasoning about the specification. It facilitates the process of evaluating designs or alternative specifications of the system-to-be. One of them is the framework proposed by Giorgini et al. [*10*] that includes AND/OR relationships among goals, but also allows one to define qualitative goal relationships, named *contribution,* to describe how much an operationalization contributes to meet a goal. In addition to this qualitative formalization, this framework consists in a label propagation algorithm and quantitative semantics for the new relationships.

Cysneiros and Yu have also proposed [*6*] a framework with high power expressive for dealing with NFRs working with them from the early stages of the software development. This framework considers NFR as goals that might conflict among each other and must be represented as softgoals to be satisfied. The softgoal concept was introduced to cope with the abstract and informal nature of NFR. The softgoals decomposition and treatment are similar to the above mentioned goals.

B. *Use Cases*

They are perhaps one of the most popular approaches to requirements specification. They have been widely embraced by the industrial community due to their straightforward notation and application. These properties allow stakeholders to easily understand them, and this contributes to the elicitation and validation of the requirements. Another factor that denotes their popularity is that Use Cases are the only notation included in UML for requirement modeling.

According to Cockburn [*4*], a use case captures a contract between the stakeholders of a system about its behavior. The use case describes the system's behavior under various conditions as it responds to a request from one of the stakeholders, called the primary actor. The primary actor initiates an interaction with the system to accomplish a goal. The system responds, protecting the interests of all the stakeholders. Different sequences of behavior, or scenarios, can unfold, depending on the particular requests made and conditions surrounding the requests. The use case collects together those different scenarios.

The relationships between different use cases and actors are shown in a UML use cases diagram (see Figure 2) [*11*]. In this kind of diagrams, we find four types of relationships:

- *Include*: a directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case.
- *Extend*: the behavior of the extension use case may be inserted into the extended use case under some conditions.
- *Generalization*: a given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case.
- *Association*: an association exists whenever an actor is involved in an interaction described by a use case.

Using these diagrams, we are able to represent functional requirements. Nevertheless, the use case diagram proposed by Booch et al [*11*] does not provide us with enough expressiveness for non-functional requirements, being a text template called *Supplementary Specification* the only available alternative. This lack of expressiveness has several associated problems such as loss of traceability during the software development process because no automatic support is provided to establish the relation between use cases and NFR.

C. *Viewpoints*

In this approach, the system-to-be is defined according to the context where it is going to perform its main computation. With this aim, it is defined considering all the involved stakeholders and assigning a different *viewpoint* to each party. Each viewpoint is a model that encapsulates a partial knowledge about the system-to-be and the domain, specified in a particular, suitable representation scheme [*5*]. Therefore, the system requirements are described by means of a combination of viewpoints, each one created by a different person implicated in the design process. An example of a viewpoint represented by a Petri Net can be seen in Figure 7.

There is not a standard notation that can be used to describe Viewpoints but every proposal identifies different concepts as relevant for the description of a Viewpoint. One of the most accepted proposals is the one presented by Nuseibeh et al. that entails five slots [*12*]:

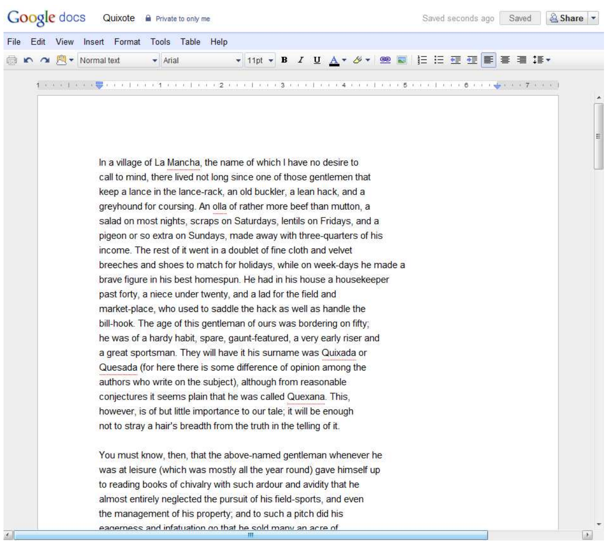
- *Style slot*: description of the representation scheme used by the viewpoint.
- *Work plan slot*: description of the development actions, process and strategy of the viewpoint.
- *Domain slot*: identifies the area of concern of the viewpoint with respect to the overall system under development.
- *Specification slot*: describes the viewpoint domain in the notation described in the style slot.
- *Work record*: maintains the development state and history of the viewpoint specification (in terms of the work plan actions performed)

One of the main problems of this approach is that it allows each stakeholder to use a different that notation more appropriate to the domain that the viewpoint belongs to. However, to avoid the great ambiguity inherent to this open representation, Finkelsetin et al. [*5*] propose the use of templates based on empty viewpoints that have some slots partially filled in. Specifically, they have defined the following templates: (i) Functional Hierarchy; (ii) System Block Diagram; (iii) Action Tables; (iv) Object Structure; (v) Petri Net.

The main advantage this approach has is the facility to find out conflicts between requirements stated by different stakeholders. In addition, some notations provide support for bottom-up and top-down traceability. For instance, Nuseibeh et al. [*12*] use the *work record* (Figure 3) to document every action or process the viewpoint has suffered throughout its history. They also provide specific notation to deal with non-functional requirements.

III. CASE STUDY

As case study to assess how these requirement specification techniques perform for collaborative system Google Docs [7] (see Figure 1) was used. Google Docs is a free, Web-based word processor, spreadsheet, presentation, form, and data storage service provided by Google. It allows users to create and edit documents online collaborating in real-time with other users remotely located via the Internet. Google Docs serves as a collaborative tool for editing documents so that they can be shared, opened, and edited by multiple users at the same time. This system was selected for our analysis because it is widely-known and it features a clear collaborative focus as its main goal.



In a village of La Mancha, the name of which I have no desire to
call to mind, there lived not long since one of those gentlemen that
keep a lance in the lance-rack, an old buckler, a lean hack, and a
greyhound for coursing. An olla of rather more beef than mutton, a
salad on most nights, scraps on Saturdays, lentils on Fridays, and a
pigeon or so extra on Sundays, made away with three-quarters of his
income. The rest of it went in a doublet of fine cloth and velvet
breeches and shoes to match for holidays, while on week-days he made a
brave figure in his best homespun. He had in his house a housekeeper
past forty, a niece under twenty, and a lad for the field and
market-place, who used to saddle the hack as well as handle the
bill-hook. The age of this gentleman of ours was bordering on fifty;
he was of a hardy habit, spare, gaunt-featured, a very early riser and
a great sportsman. They will have it his surname was Quixada or
Quesada (for here there is some difference of opinion among the
authors who write on the subject), although from reasonable
conjectures it seems plain that he was called Quexana. This,
however, is of but little importance to our tale; it will be enough
not to stray a hair's breadth from the truth in the telling of it.

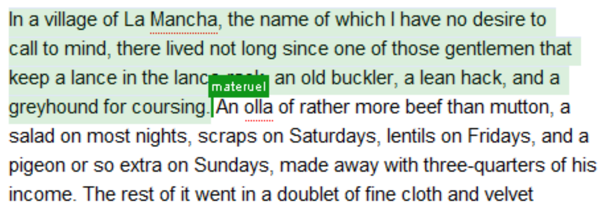
You must know, then, that the above-named gentleman whenever he
was at leisure (which was mostly all the year round) gave himself up
to reading books of chivalry with such ardour and avidity that he
almost entirely neglected the pursuit of his field-sports, and even
the management of his property; and to such a pitch did his
eagerness and infatuation go that he sold many an acre of

Figure 1. Google Docs interface

As a starting point for our evaluation of the requirements techniques, we identified those workspace awareness techniques implemented in Google Docs from the set of techniques proposed by Gutwin [*13*]. These techniques, which are commented in the following subsections, can be found also as patterns for user collaboration in [*14*].

A. *Telepointers*

This technique allows us to be aware of the other user's cursor position and whether they have selected a text fragment or not (see Figure 2). When a remote user is writing, we can realize it in real-time. Close to the cursor the user's nickname appears overlapped with the text. In addition, if the user selects some text, it is highlighted by marking it with the user's color.

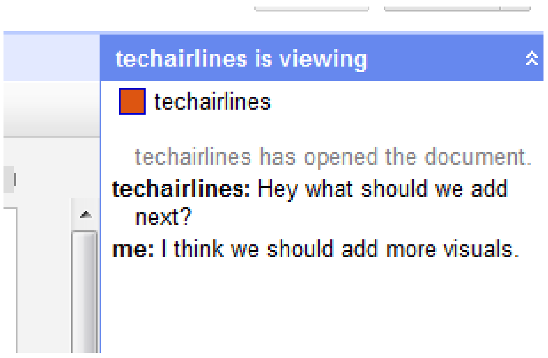


In a village of La Mancha, the name of which I have no desire to
call to mind, there lived not long since one of those gentlemen that
keep a lance in the lanc[illegible] an old buckler, a lean hack, and a
greyhound for coursing. An olla of rather more beef than mutton, a
salad on most nights, scraps on Saturdays, lentils on Fridays, and a
pigeon or so extra on Sundays, made away with three-quarters of his
income. The rest of it went in a doublet of fine cloth and velvet

Figure 2. Remote cursor and remotely selected text fragment

B. *Avatars*

Google Docs does not implement avatars itself. Instead it shows a list of participants that are editing simultaneously the same document (see Figure 3). By using this list, users can communicate with each other by using the chat, which can be shown or hidden at any time. In addition, by using this chat view, users can notice the color assigned to any of her/his collaborators.

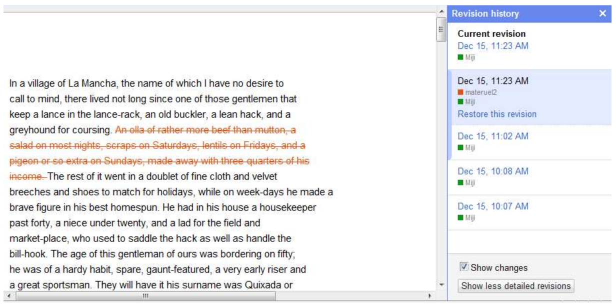


techairlines has opened the document.
techairlines: Hey what should we add next?
me: I think we should add more visuals.

Figure 3. Two users chatting through the participant list

C. *Expressing information about authorship / about the past*

These two awareness techniques are used to make available to the users the history of changes carried out. They have been implanted by Google Docs using a *revision history*. It allows the system to keep track of all the changes made by the users to the different types of documents being edited (see Figure 4). This revision history provides a mean for users to review the changes made to the documents. In this revision history the changes made by each user are denoted by using different colors. In addition, if the change made is a deletion, then the text will be also in strikethrough style. This functionality can be activated or deactivated at anytime. This revision history has two levels of detail, depending on the amount of information shown in it. The user may switch between these two levels of detail at anytime.



In a village of La Mancha, the name of which I have no desire to
call to mind, there lived not long since one of those gentlemen that
keep a lance in the lance-rack, an old buckler, a lean hack, and a
greyhound for coursing. ~~An olla of rather more beef than mutton, a
salad on most nights, scraps on Saturdays, lentils on Fridays, and a
pigeon or so extra on Sundays, made away with three quarters of his
income.~~ The rest of it went in a doublet of fine cloth and velvet
breeches and shoes to match for holidays, while on week-days he made a
brave figure in his best homespun. He had in his house a housekeeper
past forty, a niece under twenty, and a lad for the field and
market-place, who used to saddle the hack as well as handle the
bill-hook. The age of this gentleman of ours was bordering on fifty;
he was of a hardy habit, spare, gaunt-featured, a very early riser and
a great sportsman. They will have it his surname was Quixada or

Current revision
Dec 15, 11:23 AM
🟩 Miji

Dec 15, 11:23 AM
🟥 materuel2
🟩 Miji
Restore this revision

Dec 15, 11:02 AM
🟩 Miji

Dec 15, 10:08 AM
🟩 Miji

Dec 15, 10:07 AM
🟩 Miji

✓ Show changes

Figure 4. Revision story showing text elimination

IV. Empirical Evaluation

To evaluate the different RE techniques mentioned in section III, we are going to model the above mentioned awareness features using each one of these techniques. First, we have to distinguish what Google Docs characteristics can be modeled by using a functional or non-functional requirement. The *telepointer* and *avatar* techniques result in NFRs because they contribute to increase some quality in use, such as usability. Nevertheless, the third characteristic (*Expressing information about authorship / about the past*), despite contributing positively to the above mentioned quality features, it should be considered functional, due to the historical information storage and the rollback function. In addition, we have also associated the awareness functionalities both with the three characteristics of the CSCW systems (collaboration, communication and coordination) and, with the characteristics of the ISO/IEC 25010, Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model [*15*]. This standard will help us to organize properly the specification of the system following the recommendations of Moreira et al.[*16*]. The evaluation is presented next following the chronological order it was carried out. First, in section A it is described how the case study was modeled applying the three approaches. Second, in section B the main results of the evaluation are presented.

A. Modeling the Case Study

After analyzing the characteristics of Google docs described in section III, and according to the Gutwin's framework for collaborative systems, we have specified the systems FRs (Table 3 illustrates a partial description of the system).Next, as can be observed in Table 4, each awareness functionality detected in the system was related to some quality factors of the SQuaRE standard in order to identify the NFR of Google Docs. We would like to highlight that here we are describing partially the requirements of Google docs to facilitate the understandability of the evaluation.

TABLE 3. RELATION BETWEEN AWARENESS ELEMENTS AND FRS

Category	Element	Functional Requirement
Who	Presence	Know who is participating
What Where	Action Location	See other user's actions
Who When	Authorship Event history	Keep the changes' authorship

TABLE 4. RELATION BETWEEN QUALITY FACTORS AND AWARENESS FUNCTIONALITIES

Quality Factor	Awareness Functionality
Functional Suitability	Revision History Telepointers Participant List
Reliability	Revision History
Performance Efficiency	Telepointers
Operability	Telepointers Participant List
Security	Revision History

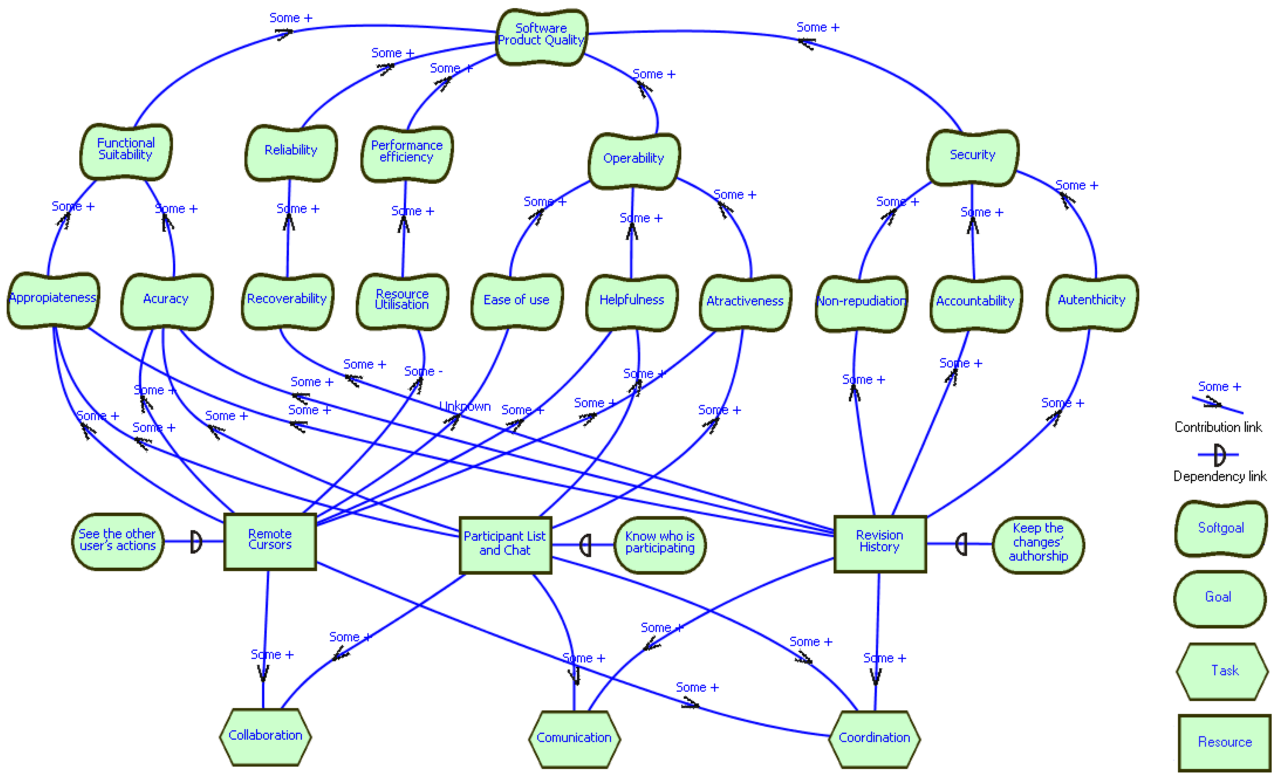


Figure 5. Goal-Oriented model

1) Goal-Oriented requirement specification

In order to carry out the specification of Google Docs, the *i** notation [*17*] was used, as it is the most widely known and accepted proposal in Goal-Oriented. Using this notation, we specified each one of the SQuaRE quality factors [*15*], previously identified in Table 4, as root softgoals of the system as shown in Figure 5. These softgoals were refined into other softgoals selecting those quality factors of SQuaRE standard more appropriate for the system. Each one of the awareness functionalities were specified as resources provided by the system that contribute positively to satisfy some of the softgoals, that is, some quality factors. However, it can be noticed that also some of them contribute negatively because the constraints they impose. This is the case of remote cursors, because they increase the resource utilization. Moreover, the ease of use depends, among other factors, on the user's experience with this kind of systems. In addition, the three FR identified in Table 3 have been specified as goals of the system that have dependency relationships with the resources. We have also specified how the awareness techniques contribute positively to the functional aspects of collaborative systems specified as tasks in the goal model.

2) Use Cases

In order to represent the system using this RE technique, we have come up against a problem: the lack of expressiveness of use case diagrams (as defined in UML [*11*]) to describe NFR. Therefore, if this notation is used the only alternative is to describe the identified FR, as shown in Figure 6(a), and exploit a different document, such as the supplementary specification, for the NFR. As was stated above, this alternative presents some limitations, such as the poor support for traceability between FRs and NFRs.

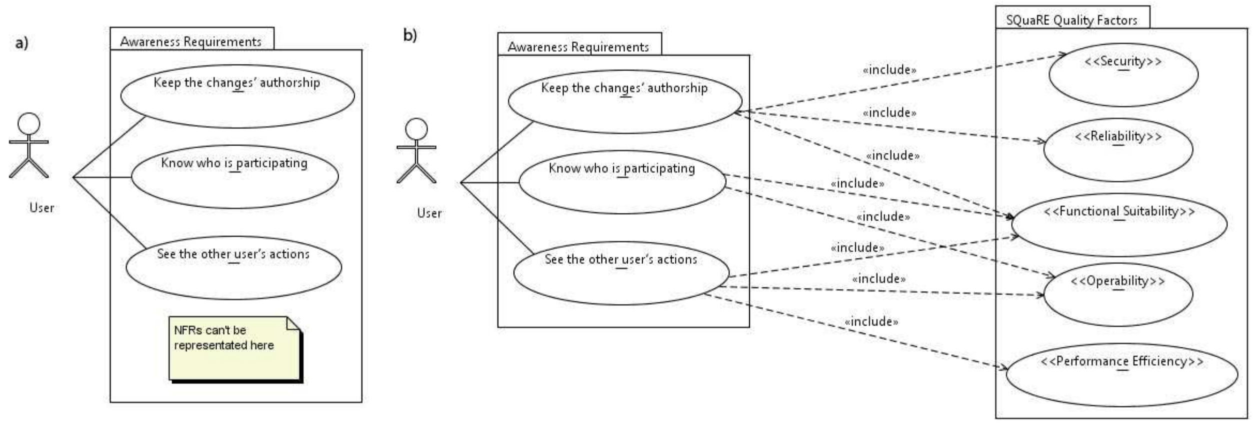

a)
Awareness Requirements
Keep the changes' authorship
Know who is participating
See the other user's actions
User
NFRs can't be representated here
b)
Awareness Requirements
Keep the changes' authorship
Know who is participating
See the other user's actions
User
SQuaRE Quality Factors
<<Security>>
<<Reliability>>
<<Functional Suitability>>
<<Operability>>
<<Performance Efficiency>>
«include»

Figure 6. Use Case Diagram with extension for NFRs

In order to overcome the lack of expressiveness of the Use Cases, we have decided to exploit the extension proposed by Moreira et al. [*18*]. This extension allows one to describe some stereotypes to describe quality factors such as security or reliability that can be applied to use cases to specify NFR. Using this extension the system requirements were specified as shown in Figure 6 (b). As can be observed, this alternative does allow us to describe NFR and trace them properly.

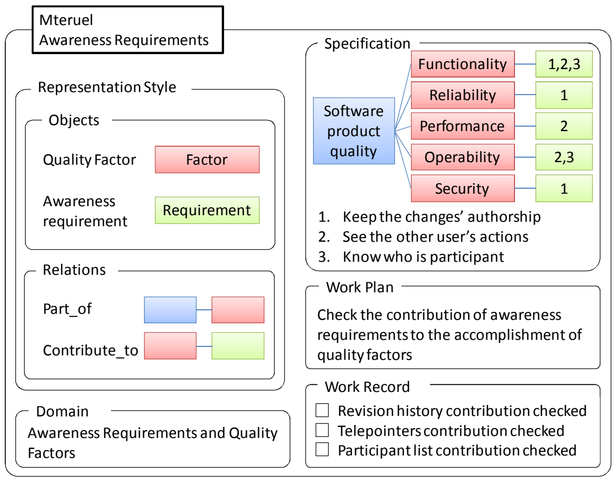
3) Viewpoints

In order to apply this RE technique, it was necessary to define firstly the representation style to specify the viewpoints of the system. For its definition, it was considered a must that this style allows us to relate the awareness functionalities to the quality factors. The decision made was to define a viewpoint representation style formed by two objects (quality factors and awareness requirements) and two relations between that objects (compositions and contributions). In addition, it was necessary to define the domain for which to apply the technique, that is, *awareness techniques and quality factors*.

Taking into account the representation style, the quality engineer's viewpoint was defined as illustrated in Figure 7. As can be observed, the quality factors were specified by means of a tree whose leaves are the awareness techniques of GoogleDocs. In this way, it can be established a direct relationships between the quality factors and the functionality of the system.

B. Evaluating RE techniques

Using as input the different specifications of the system, the evaluation of the different RE techniques was carried out by using *DESMET* [*19*]. It is a set of techniques applicable for evaluating both Software Engineering methods and tools. Specifically, we have used the method based on qualitative case study that describes a feature-based evaluation. Following the guidelines of this technique, we have prepared an initial list of features that a RE technique for collaborative systems must accomplish, as described in Table 5. As can be observed, some features are directly related to the specification of NFRs.



Mteruel
Awareness Requirements
Representation Style
Objects
Quality Factor
Factor
Awareness requirement
Requirement
Relations
Part_of
Contribute_to
Domain
Awareness Requirements and Quality Factors
Specification
Software product quality
Functionality
Reliability
Performance
Operability
Security
1,2,3
1
2
2,3
1
1. Keep the changes' authorship
2. See the other user's actions
3. Know who is participant
Work Plan
Check the contribution of awareness requirements to the accomplishment of quality factors
Work Record
Revision history contribution checked
Telepointers contribution checked
Participant list contribution checked

Figure 7. Viewpoint for Awareness Requirements and Quality Factors

TABLE 5. LIST OF FEATURES FOR MODELS EVALUATION

Feature	Description
Awareness Representation	The model should allow one to represent the awareness characteristics of the system
Quality Factors Representation	The model must represent the SQuaRE characteristics and sub-characteristics
NFR Representation	The model should be able to represent NFRs graphically
Hierarchical Representation	The relation between the model elements should be hierarchical
Standard Representation	The model must be based on a widely extended standard representation
Model Complexity	The model complexity should not be too high
Quantitative Model	The model must allow one to quantify the relations between represented elements
Traceability	The represented requirements should be traceable throughout the software development process

Once the previous table is filled in, DESMET establishes that an importance degree should be assigned to each identified feature. Specifically, the degrees to apply are:

- M: Mandatory
- HD: Highly Desirable
- D: Desirable
- N: Nice to have

Using these degrees Table 6 was filled in. As can be noticed, the most important features to be supported are both the NFR representation and traceability required by collaborative systems.

TABLE 6. IMPORTANCE OF THE FEATURES

Feature	Importance
Awareness Representation	M
Quality Factors Representation	M
NFR Representation	HD
Traceability	HD
Quantitative Model	D
Hierarchical Representation	D
Standard Representation	D
Model Complexity	N

Next, we have established a scale to evaluate each one of the described features. Specifically, we have used the scale proposed in DESMET that has been described in Table 7. This scale was employed to evaluate each feature according to the following factors:

- CAT: Conformance Acceptability Threshold.
- CSO: Conformance score obtained for candidate method.

Once each feature was evaluated, the difference between CAT and CSO factors was computed as shown in the column Difference (Dif) in Tables 8, 9, 10 and 11.

TABLE 7. JUDGEMENT SCALE TO ASSESS TOOL SUPPORT FOR A FEATURE

Generic scale point	Definition of Scale point	Scale Point Mapping
Makes things worse	Cause Confusion. The way the feature is implemented makes it difficult to use and/or encouraged incorrect use of the feature	-1
No support	Fails to recognise it. The feature is not supported nor referred to in the user manual	0
Little support	The feature is supported indirectly, for example by the use of other tool features in non-standard combinations.	1
Some support	The feature appears explicitly in the feature list of the tolls and user manual. However, some aspects of feature use are not catered for.	2
Strong support	The feature appears explicitly in the feature list of the tolls and user manual. All aspects of the feature are covered but use of the feature depends on the expertise of the user	3
Very strong support	The feature appears explicitly in the feature list of the tools and user manual. All aspects of the feature are covered and the tool provides tailored dialogue boxes to assist the user.	4
Full support	The feature appears explicitly in the feature list of the tolls and user manual. All aspects of the feature are covered and the tool provides user scenarios to assist the user such as "Wizards".	5

Next, we should highlight that a variation of the DESMET method has been used. Specifically, the importance (Imp) of each feature has been weighted in a scale from 1 to 4 (Nice to have – 1, Desirable – 2, Highly

Desirable– 3, Mandatory – 4). The importance was used to compute the final score of each feature by multiplying the Importance by the Difference. This calculation is shown in the column Score (Sco) in Tables 8, 9, 10 and 11. Lastly, the final score of each technique (Total) was obtained by adding the scores of all the features. This framework has been used to evaluate the different RE techniques studied. Note that two evaluations have been performed for the Use Case technique depending on whether it is able to represent NFRs or not.

TABLE 8. EVALUATION FOR GOAL-ORIENTED REQUIREMENT SPECIFICATION

Feature	Imp	CAT	CSO	Dif	Sco
Awareness Representation	4	5	5	0	0
Quality Factors Representation	4	4	5	1	4
NFR Representation	3	3	5	2	6
Traceability	3	3	3	0	0
Quantitative Model	2	2	1	-1	-2
Hierarchical Representation	2	2	3	1	2
Standard Representation	2	2	2	0	0
Model Complexity	1	1	1	0	0
Total					**10**

TABLE 9. EVALUATION FOR USE CASE

Feature	Imp	CAT	CSO	Dif	Sco
Awareness Representation	4	5	2	-3	-12
Quality Factors Representation	4	4	0	-4	-16
NFR Representation	3	3	1	-2	-6
Traceability	3	3	1	-2	-6
Quantitative Model	2	2	0	-2	-4
Hierarchical Representation	2	2	0	-2	-4
Standard Representation	2	2	5	3	6
Model Complexity	1	1	3	3	3
Total					**-39**

TABLE 10. EVALUATION FOR USE CASE WITH NFRS REPRESENTATION

Feature	Imp	CAT	CSO	Dif	Sco
Awareness Representation	4	5	3	-2	-8
Quality Factors Representation	4	4	4	0	0
NFR Representation	3	3	3	0	0
Traceability	3	3	3	0	0
Quantitative Model	2	2	0	-2	-4
Hierarchical Representation	2	2	1	-1	-2
Standard Representation	2	2	3	1	2
Model Complexity	1	1	2	1	1
Total					**-11**

TABLE 11. EVALUATION FOR VIEWPOINTS

Feature	Imp	CAT	CSO	Dif	Sco
Awareness Representation	4	5	1	-4	-16
Quality Factors Representation	4	4	1	-3	-12
NFR Representation	3	3	0	-3	-9
Traceability	3	3	1	-2	-6
Quantitative Model	2	2	0	-2	-4
Hierarchical Representation	2	2	2	0	0
Standard Representation	2	2	0	-2	-4
Model Complexity	1	1	1	0	0
Total					**-51**

Figure 8 shows graphically the scores obtained by each one of the RE techniques. As can be observed, the Goal-Oriented approach is the only one that has a positive score. Despite this positive score, it has been negatively evaluated for the Quantitative Model feature as *i** only provides a partial support for quantifying the relations among requirements when using contribution links. The use case technique fails basically to describe the awareness model, since it does not support NFR. It also fails in the Quantitative model as it does not provide any assistance in this sense. Finally, it neither provides support for hierarchical representation. The limitation for NFR is overcome when UC-NFR is used, however, it does provide no improvement for the other two shortcomings. Finally, the less suitable technique to this problem is the Viewpoint, as it lacks enough support for most of the analyzed features.

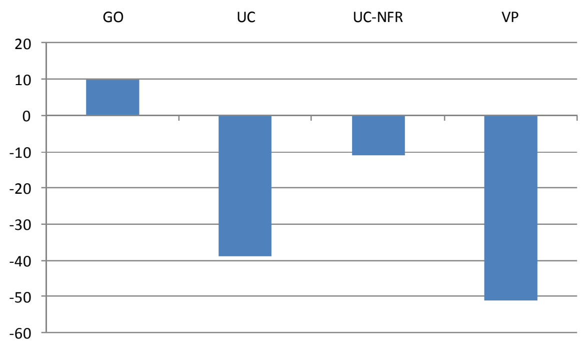


Figure 8. Empirical analysis results

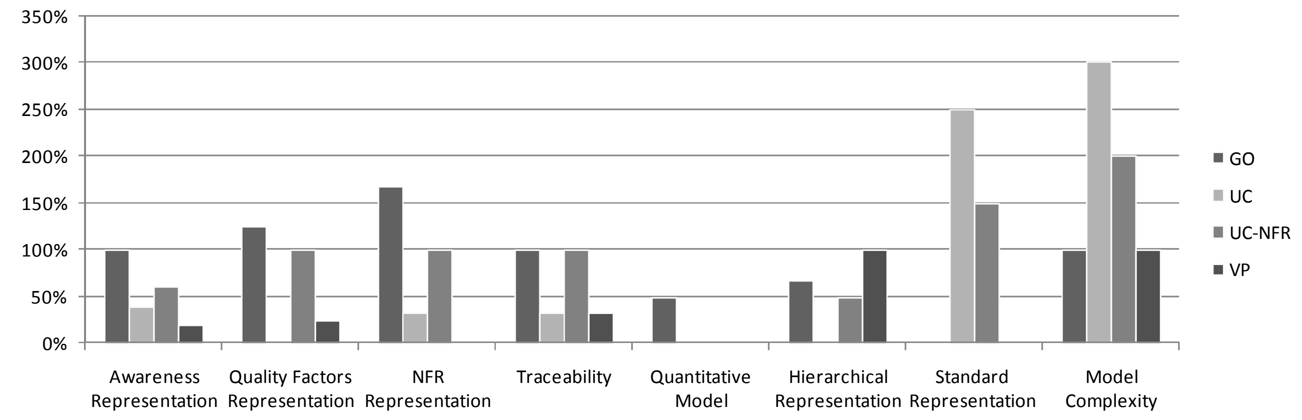


Figure 9. Results relative to distinct features

In addition, as DESMET suggests, we have performed a comparative of the percentage of each feature satisfied by each RE technique analyzed as shown in Figure 9. It should be highlighted that the no one of the analyzed models is too complex what allows stakeholders to convey the information of the system in an easier and more intuitive way. Use Case technique stands out from the other techniques for the standardization of its notation. Regarding to the Hierarchical representation, it can be stated that the analyzed techniques, except for the Use Cases, provide some kind of support what helps to manage the complexity of the specification. It is worth noting that the Goal-Oriented technique is the only one that has some support for Quantitative Model, despite the relevance this feature should have, as it helps to analyze the requirement specification. Regarding traceability feature, Use Case technique is the most salient one because of its integration in the Rational Unified Process [*20*]. Both Goal-Oriented and UC-NFR stand out in the representation of NFRs and quality factors. NFR representation is a must for the specification of collaborative systems. Unfortunately, Awareness representation is poorly supported by all the Requirements Engineering techniques analyzed, despite being the most important feature for the specification of collaborative systems. In the light of analysis of the results, Goal-Oriented technique seems to be the most promising approach to the specification of collaborative systems.

V. CONCLUSIONS AND FURTHER WORK

After this empirical experiment, we can conclude that the analyzed techniques are not fully appropriate to model the awareness requirements for a collaborative system. In fact, the only technique that obtained a positive result in our empirical analysis was the Goal-Oriented requirement specification.

These results support our initial hypothesis: a Requirement Engineering technique to address the problems detected during study is required. In this sense, the Goal-Oriented technique seems to be the most promising as it does provide support for NFR and a Quantitative Model, in addition to its facilities to trace properly both FR and NFR. However, it does exhibit some shortcomings for dealing with awareness representation that should be addressed before it is applied for the specification of collaborative systems. This constitutes one of our future and challenging works: to adapt/extend this notation for this kind of systems.

In addition, another future work is the definition of techniques that allow us to check that the defined model can be used for validation purposes. That is, its conformance with the SQuaRE Quality in Use factors (usability, flexibility and safety) [*15*] should be evaluable in an easy and intuitive way, once the system is fully developed.

ACKNOWLEDGEMENTS

This work has been partially supported by grants (PEII09-0054-9581) from the Junta de Comunidades de Castilla-La Mancha and also by a grant (TIN2008-06596-C02-01) from the Spanish Government.

REFERENCES

[1] José Luis Garrido, "AMENITIES: una Metodología para el Desarrollo de Sistemas Cooperativos Basada en Modelos de Comportamiento y Tareas," University of Granada, 2003.

[2] Carl Gutwin and Saul Greenberg, "A Descriptive Framework of Workspace Awareness for Real-Time Groupware," *Computer Supported Cooperative Work*, 2002.

[3] Hike Hochmuller, "Towards the Proper Integration of Extra-Functional Requirements," *Australasian Journal of Information Systems*, vol. 6, no. 2, 1999.

[4] Alistair Cockburn, *Writting Effective Use Cases.*: Addison-Wesley, 2000.

[5] Anthony Finkelsetin, Jeff Kramer, Bashar Nuseibeh, L. Finkelstein, and Michael Goedicke, "Viewpoints: A Framework for Integrating Multiple Perspectives in System Development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, no. 1, pp. 31-57, 1992.

[6] Luiz Marcio Cysneiros and Eric Yu, "Non-Functional Requirements Elicitation," in *Perspectives on Software*

Requirements, Julio Cesar Sampaio do Prado Leite and Jorge Horacio Doorn, Eds.: Springer, 2003, ch. 6.

[7] Google Inc. (2010) Google Docs. [Online]. http://docs.google.com

[8] Axel van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," in *Proceedings 5th IEEE International Symposium on RE*, Toronto, 2001, pp. 249-263.

[9] Evangelia Kavakli and Pericles Loucopoulos, "Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods," *Information Modeling Methods and Methodologies*, pp. 102-124, 2005.

[10] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani, "Formal Reasoning Techniques for Goal Models," *Journal on Data Semantics*, vol. 2800/2003, pp. 1-20, 2004.

[11] Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*.: Addison Wesley, 2005.

[12] Bashar Nuseibeh, Jeff Kramer, and Anthony Finkelstein, "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification," *IEEE Transactions on Software Engineering*, vol. 20, no. 10, 1994.

[13] Carl Gutwin, "Workspace Awareness in Real-Time Distributed Groupware," in *HCI '96 Proceedings of HCI on People and Computers XI*, 1996, pp. 281-298.

[14] Till Schümmer and Stephan Lukosch, *Patterns for Computer-Mediated Interaction*.: John Wiley & Sons Ltd, 2007.

[15] Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model.

[16] Ana M. D. Moreira, João Araújo, and Awais Rashid, "A Concern-Oriented Requirements Engineering Model," in *CAiSE*, 2005, pp. 293-308.

[17] Xavier Franch, "On the Lightweight Use of Goal-Oriented Models for Software Package Selection," in *CAiSE*, 2005, pp. 551-566.

[18] Ana Moreira, João Araújo, and Isabel Brito, "Crosscutting Quality Attributes for Requirements Engineering," in *SEKE*, 2002, pp. 167-174.

[19] Barbara Kitchenham, "DESMET: A method for evaluating Software Engineering methods and tools," Department of Computer Science, University of Keele, 1996.

[20] Philippe Kruchten, *The Rational Unified Process: An Introduction*, 2nd ed.: Addison Wesley, 2000.