A COMPARATIVE OF GOAL-ORIENTED APPROACHES TO MODELLING REQUIREMENTS FOR COLLABORATIVE SYSTEMS

Miguel A. Teruel, Elena Navarro, Víctor López-Jaquero, Francisco Montero and Pascual González
LoUISE Research Group, Computing Systems Department, University of Castilla - La Mancha, Albacete, Spain
MiguelAngel.Teruel@uclm.es, Elena.Navarro@uclm.es, victor@dsi.uclm.es, fmontero@dsi.uclm.es, pgonzalez@dsi.uclm.es

Keywords: Goal-Oriented, KAOS, NFR, *i**, Collaborative Systems, CSCW, Awareness, Requirements Engineering, Non-Functional Requirements, Quality.

Abstract: A collaborative system is a software allowing 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. However, when these systems have to be specified for development severe difficulties emerge to describe the requirements associated to these special functionalities, usually considered non-functional requirements. Therefore, the selection and use of proper requirements engineering techniques becomes a challenging and important decision. In this paper three Goal-Oriented approaches, namely NFR framework, *i** and KAOS, are evaluated in order to determine which one is the most suitable to deal with this problem of requirements specification in collaborative systems.

1 INTRODUCTION

A collaborative system (a.k.a. Computer Supported Cooperative Work system, CSCW system) is a software whose users can perform collaboration, communication and coordination tasks. Unlike a conventional single-user system, a CSCW system has to be specified by using a special set of requirements of non-functional nature. These requirements usually result from the users' need of being aware of the presence and activity of other remote users with whom they perform the above mentioned collaborative tasks, that is, the *Workspace Awareness* (WA).

Workspace Awareness 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* (Gutwin and Greenberg, 2002). Gutwin et al. presented a conceptual framework to establish what information makes up workspace awareness. This information is obtained by answering the questions "who, what and, where" (see Table 1). That is, when we work with others users 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 happen.

Table 1: Elements of Workspace Awareness.

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?
	Artefact	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?

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) (Hochmuller, 1999). 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 a previous work (Teruel et al., 2011) it was analyzed which technique, Goal-Oriented (GO), Use Cases or Viewpoints is more appropriate to specify the requirements of collaborative systems and it was determined that GO provides more facilities for this kind of systems. In this paper, we study the applicability of three Goal Oriented (GO) approaches (NFR Framework (Cysneiros and Yu, 2003), *i** Framework (Castro, Kolp and Mylopoulos, 2001) and KAOS Methodology (van Lamsweerde, 2001)) for the specification of collaborative systems, paying special attention to the awareness requirements. In order to carry out this study, the awareness requirements of a real system (Google Docs (Google, 2011)) were specified. After modelling the system, an empirical analysis was conducted in order to compare these different techniques goal-oriented techniques.

This paper is structured as follows. After this introduction, in section 2, the selection of GO techniques for modelling this kind of systems is justified. In section 3, three GO approaches applicable to awareness requirements for collaborative systems are analysed. In section 4, an example of a widely known collaborative system is presented: Google Docs. In section 5, an empirical evaluation of the previous techniques for modelling awareness requirements in Google Docs is presented. Finally, some conclusions and future works round up this work.

2 RELATED WORKS

This paper is a follow-up of the work presented in (Teruel et al., 2011), where we analysed different Requirement Engineering techniques applied to collaborative systems. The main result of this evaluation was that the most appropriate technique for this kind of systems is Goal Oriented (GO). Nevertheless, in (Teruel et al., 2011) the evaluation did not focus on a specific GO proposal.

In the context of Requirements Engineering, the GO approach (van Lamsweerde, 2001) has proven its usefulness for eliciting and defining requirements. More traditional techniques, such as Use Cases (Cockburn, 2000), only focus on establishing the features (i.e. activities and entities) that the system-to-be should support. Nevertheless, GO proposals focus on why systems are being constructed by providing the motivation and rationale to justify the software requirements specification. They are not only useful for analyzing goals, but also for elaborating and refining them.

A GO model can be specified in a variety of formats, by using a more or less formally defined notation. These notations can be informal, semi-informal or formal approaches. Informal approaches generally use natural language to specify goals; semi-formal use mostly box and arrow diagrams; finally, in formal approaches goals are expressed as logical assertions in some formal specification language (Kavakli and Loucopoulos, 2004). No matter its formality, a goal model is built as a directed graph by means of a refinement of the systems goals. 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 (Kavakli and Loucopoulos, 2004) 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 (Teruel et al., 2011)(Chung et al., 2000). In this work, three different GO proposals are used to model the requirements of a collaborative system: Google Docs. This system will allow us to evaluate which proposal is the most useful to describe the requirements of the so called workspace awareness.

3 GOAL ORIENTED PROPOSALS: AN ANALITICAL BACKGROUND

This section presents briefly the GO proposals, NFR, *i** and KAOS, analyzed to determine which one is the most appropriate for specifying collaborative systems. They are used in section 5 to describe the running example in order to perform the evaluation.

3.1 NFR Framework

This GO proposal was proposed by (Cysneiros and Yu, 2003) and aims at dealing with Non-Functional Requirements (NFRs), also known as Quality Requirements. Unlike Functional Requirements, NFRs specify constraints for the system, as well as particular notions of *quality factors* a system should meet, such as, accuracy, usability, safety, performance, reliability or security. Hence, it can be stated that while functional requirements describe "what" the system will do, NFRs constraint "how" the system will accomplish the "what". As a consequence, NFRs are always linked to a Functional Requirement.

To elicit NFRs, the authors propose the use of a strategy anchored in *Language Extended Lexicon* (LEL) (Sampaio and Franco, 1993). LEL is based on a controlled vocabulary system made up of *symbols* being each one of them an entry expressed in terms of *notions* and *behavioural* responses. A notion records the meaning of a symbol and its fundamental relationships to other entries. A behavioural response specifies the connotation of a symbol in the universe of discourse. Each symbol may also be represented by one or more aliases and will be classified as a *subject*, a *verb* or an *object*. Once the Lexicon is finished, it is enriched with NFRs by using a knowledge base, presented as catalogues, to guide the analyst to select the likely needed NFRs and their related operationalizations.

According to the NFR Framework, NFRs goals can conflict among them and must be represented as softgoals to be satisfied. Each softgoal is decomposed into sub-goals represented by a graph structure inspired by the *And/Or* trees used in problem solving. This decomposition is done by using contribution links. Contribution links can be categorized as either *or* contributions or *and* contributions. Contribution links allow one to decompose NFRs to the point that one can state that the operationalizations of the related NFR have been met. Operationalizations are decisions about the system to meet NFRs. The elements of the NFR GO model can be seen in Figure 1.

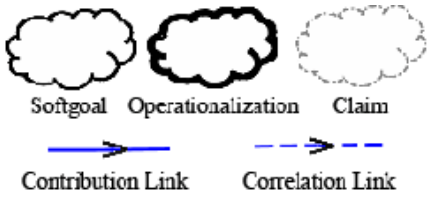

Figure 1: Elements of the NFR Framework model.

3.2 The *i** Framework

The *i** Framework (Castro, Kolp and Mylopoulos, 2001) consists in an approach for dealing with requirements in various phases of the software development process (Early and Late Requirements Analysis, Architectural and Detailed Design).

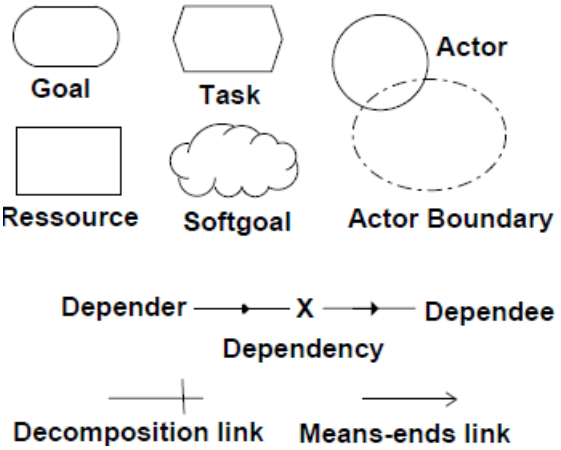

Figure 2: Elements of the *i** Framework model.

During early requirements analysis, the requirements engineer gathers and analyzes the intentions of stakeholders. These are modelled as

goals which, through some form of a goal-oriented analysis, eventually lead to the functional and non-functional requirements of the system-to-be. In *i**, early requirements are assumed to involve social actors who depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. The *i** framework includes the *strategic dependency model* for describing the network of relationships among actors, as well as the *strategic rationale model* for describing and supporting the reasoning that each actor goes through concerning its relationships with other actors. The model elements can be seen in Figure 2.

Late Requirements Analysis results in a requirements specification which describes all functional and non-functional requirements for the system-to-be. In Tropos (Mylopoulos, Castro and Kolp, 2000), a framework for requirements-driven software development, the information system is represented as one or more actors who participate in a strategic dependency model, along with other actors from the system's operational environment. In other words, the system comes into the picture as one or more actors who contribute to the fulfilment of stakeholder's goals.

During architectural design we have to select among alternative architectural styles by using as criteria the desired qualities identified earlier in the process. The analysis involves refining these qualities, represented as softgoals, to sub-goals that are more specific and more precise and then evaluating alternative architectural styles against them.

The detailed design phase is intended to introduce additional details for each architectural component of a system. To support this phase, the authors propose to adopt existing agent communication languages and message transportation mechanisms among other concepts and tools.

3.3 KAOS Methodology

The KAOS modelling language is part of the KAOS framework (van Lamsweerde, 2001) for eliciting, specifying, and analysing goals, requirements, scenarios, and responsibility assignments. A KAOS model entails six complementary views or sub-models (goal, obstacle, object, agent, operation and behaviour model) all of them related via traceability links (Pohl, 2010).

Figure 3 depicts the basic constructors for documenting agents responsibilities for goals provided by the KAOS framework. KAOS has the following elements:

- Goal: A goal describes a set of admissible system behaviors. Goals should be defined in a clear-cut manner so that one can verify whether the system satisfies a goal or not.
- Softgoal: In KAOS, softgoals are used to document preferences among alternative system behaviors. In a similar way to i*, there is no clear-cut criterion for verifying the satisfaction of a softgoal. Softgoals are hence expected to be satisfied within acceptable limits.
- Agent: While i* focuses primarily on agents within organizational structures, the agents defined in KAOS primarily relate to users and components of software-intensive systems. Therefore, an agent is defined as an active system component which has a specific role for satisfying a goal. An agent can be a human agent, a device or a software component.

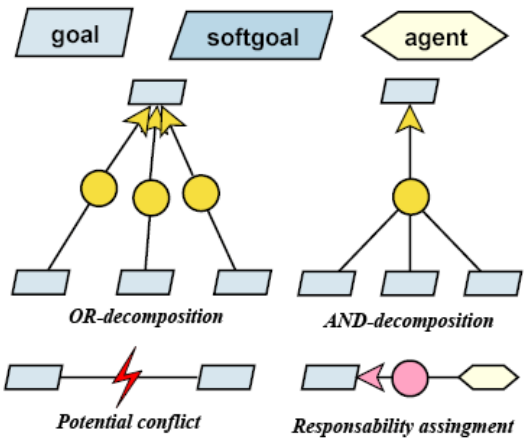

Figure 3: Basic constructs of the KAOS framework for modelling goals and assigning agents responsibilities for goals to.

Dependencies between goals are represented in the KAOS goal model by using AND/OR-decompositions and conflict links. In KAOS, goals can be assigned to agents by means of responsibility assignment links. We briefly explain these goal dependencies:

- AND/OR-decomposition: An AND-OR decomposition link relates a goal to a set of sub-goals, documenting that the goal is satisfied if all, or at least one sub-goal, is satisfied.
- Potential conflict: This link documents that satisfying one goal may prevent the satisfaction of other goal under certain conditions.
- Responsibility assignment: This link between a

goal and an agent means that this agent is responsible for satisfying the goal.

4 RUNNING EXAMPLE

As running example to assess how these GO approaches perform for collaborative system, Google Docs (Google, 2011) (see Figure 4) has been used from now on in this paper. Google Docs is a free, Web-based word processor, spreadsheet, presentation and form editor whose data storage service is provided by Google. 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.

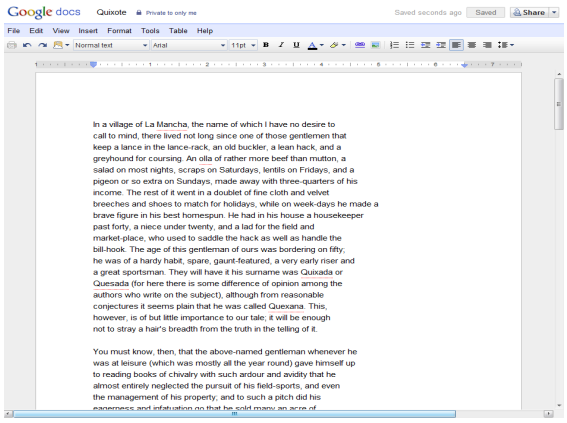

Figure 4: Google Docs interface

As a starting point for our evaluation of the requirements techniques, we identified those design solutions for awareness requirements in Google Docs from the set of techniques proposed by Gutwin (Gutwin, Greenberg and Roseman, 1996). These techniques, which are commented in the following subsections, can be found also as patterns for user collaboration in (Schümmer and Lukosch, 2007).

4.1 Remote Cursors

This technique, based in Gutwin's *telepointers* (Gutwin, Greenberg and Roseman, 1996), allows us to be aware of the other user's cursor position and whether they have selected a text fragment or not (see Figure 5). Thus, when a remote user is writing other users can notice 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 colour.

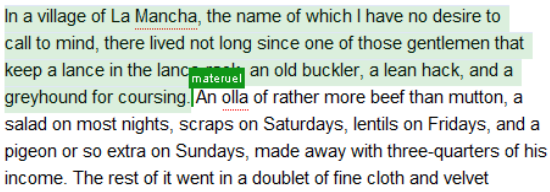

Figure 5: Remote cursor and remotely selected text fragment.

4.2 Participant List & Chat

Google Docs does not implement Gutwin's *avatar* (Gutwin, Greenberg and Roseman, 1996) technique itself. Instead it shows a list of participants that are editing simultaneously the same document (see Figure 6). By using this list, users can communicate with each other by using a chat, which can be shown or hidden at any time. In addition, by using this chat view, users can notice the colour assigned to each one of their collaborators.

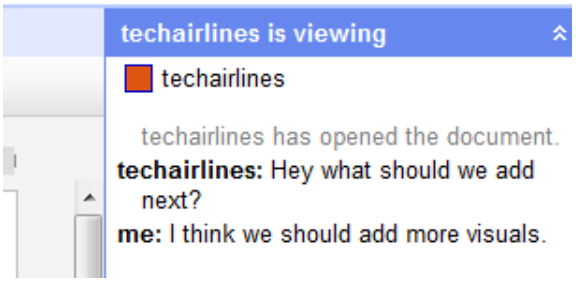

Figure 6: Two users chatting through the participant list.

4.3 Revision History

The techniques identified by Gutwin *expressing information about authorship / about the past* (Gutwin, Greenberg and Roseman, 1996) are used to make available to the users the history of changes carried out. They have been implemented by Google Docs by 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 7). 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 colours. 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 shown

information. The user may switch between these two levels of detail at anytime.

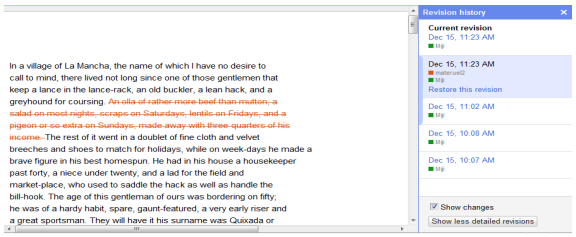

Figure 7: Revision story showing text elimination.

5 EMPIRICAL EVALUATION

To evaluate the different GO approaches mentioned in section 3, each one of the above mentioned awareness features is modelled in the following by using the different techniques. First, we have to distinguish what Google Docs characteristics can be modelled by using functional or non-functional requirements. The *telepointer* and *avatar* techniques result in NFRs because they contribute to increase some operability, such as ease of use and helpfulness. 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 collaborative 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, 2008). This standard has been used to organize properly the specification of the system following the recommendations of Moreira et al. (Moreira, Araújo and Rashid, 2005). Next, the evaluation is presented following the chronological order it was carried out. First, in section 5.1 it is described how the case study was modelled by applying the three approaches. Second, in section 5.2, the results of the evaluation are presented.

5.1 Modelling the Running Example

After analyzing the characteristics of Google docs described in section 4, and according to Gutwin's framework for collaborative systems, we have specified the systems' FRs (Table 2 illustrates a partial description of the system). Next, as can be observed in Table 3, each awareness functionality feature detected in the system has been related to some quality factors in the SQuaRE standard, in order to identify the NFRs of Google Docs. For the sake of clarity, and understanding of the evaluation, only some requirements of Google Docs are described.

Table 2: 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 3: 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

5.1.1 The NFR Framework

In this approach, the SQuaRE quality factors have been modelled by using softgoals. Nevertheless, the SQuaRE standard was used instead of the NFR collections proposed by (Cysneiros and Yu, 2003) definition to create the NFR hierarchy. Thus, it can be observed the impact that the quality sub-characteristic has on main characteristic by means of contribution links. In the same way, each characteristic contributes to achieve the software product quality (see Figure 8).

The problem here is that we are not able to represent the Functional Requirements (because this model aims only at non-functional ones), therefore the three general tasks of collaborative systems (collaboration, communication and coordination) cannot be defined. This lack of expressiveness led us to have an incomplete representation of system's requirements, so that we have to use additional models or extend this framework.

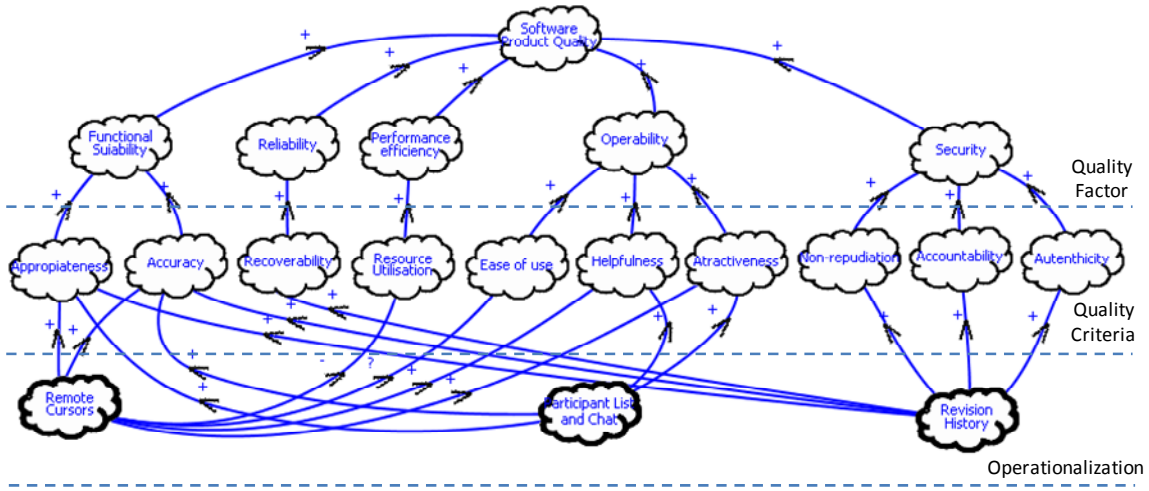

Figure 8: NFR Goal-Oriented model.

5.1.2 The *i** Framework

In order to carry out the specification of Google Docs, the *i** notation was used. Using this notation, we specified each one of the SQuaRE quality factors previously identified in Table 3, as root softgoals of the system as shown in Figure 9. These softgoals were refined into other softgoals by selecting those SQuaRE quality factors 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. It has been also specified how the awareness techniques contribute positively to the functional aspects of collaborative systems specified as tasks in the goal model.

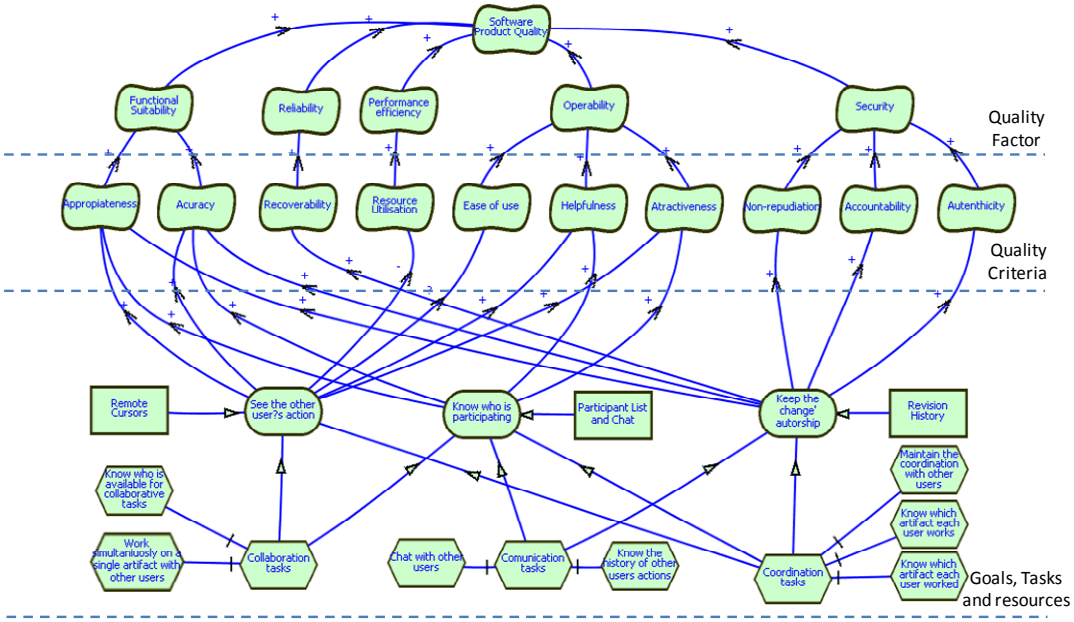

Figure 9: *i** Strategic Rationale Model.

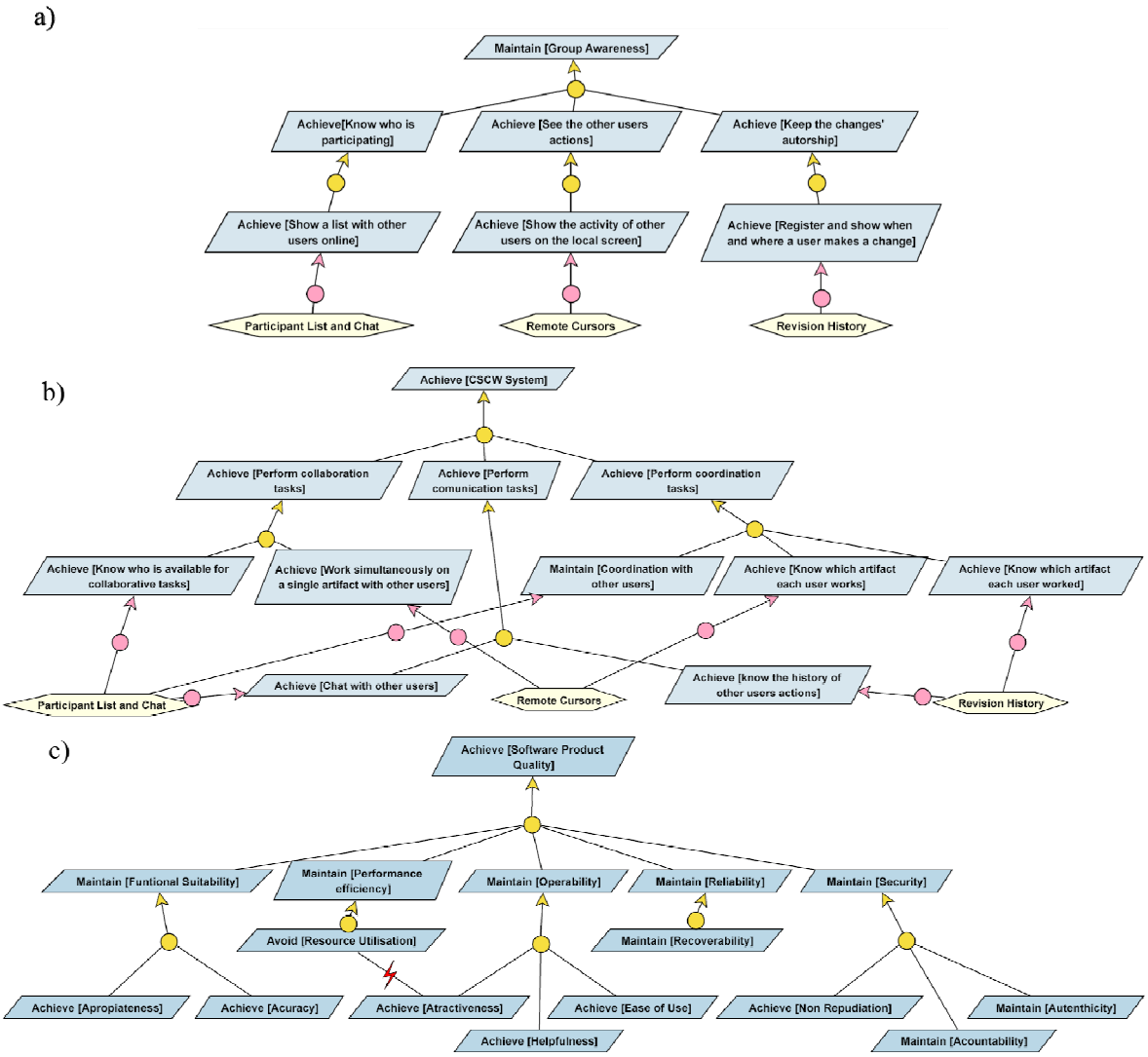

Figure 10: KAOS Goal and Responsibility Model.

5.1.3 KAOS Methodology

To model the system using this methodology, and unlike *i**, the model was decomposed in three sub-models as can be seen in Figure 10. Hence, the individual models represent (a) awareness goals, (b) collaborative systems goals and (c) software quality goals.

These diagrams (Figure 10) show three main goals and its decomposition in its sub-goals. The implemented awareness techniques have been represented here by using agents, because this element is used to represent responsibility assignment when using KAOS.

In addition, Figure 10c illustrates a potential conflict between two softgoals related to two quality sub-factors: *avoid resource utilisation* and *achieve attractiveness*. Usually, a very attractive user interface will cause a higher resource utilisation. This conflict is denoted in the graph by using a red ray.

5.2 Evaluating GO Approaches

Using as input the different specifications of the system, the evaluation of the different RE techniques was carried out by using *DESMET* (Kitchenham, 1993). It is a set of techniques applicable to evaluate both Software Engineering methods and tools. We have used the method based on a qualitative case study that describes a feature-based evaluation. Following the guidelines of this technique, an initial

Table 4: List of Features for approaches evaluation.

Feature	Description
FR and NFR Representation	The model should be able to represent graphically FR and NFRs and differentiate them
Collaborative Systems Characteristics	The model has to represent the collaboration, communication and coordination characteristics
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
Importance of Requirements	The model should represent the importance and preference between requirements
Hierarchical Representation	The relation between the model elements should be hierarchical
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

list of features was prepared that a GO approach for collaborative systems should provide (see Table 4). As can be observed, some of those features are directly related to the specification of NFRs.

Once Table 4 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

By using these degrees, Table 5 was filled in. As can be noticed, the most important features to be supported are both the NFR representation and the traceability required by collaborative systems.

Table 5: Importance of the features.

Feature	Importance
FR and NFR Representation	M
Collaborative Systems Characteristics	M
Awareness Representation	M
Quality Factors Representation	HD
Importance of Requirements	HD
Traceability	HD
Quantitative Model	D
Hierarchical Representation	D
Model Complexity	N

Next, according to DESMET, a scale to evaluate each one of the described features should be provided. The scale proposed by DESMET (see Table 6) was applied 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 7, 8 and 9.

Table 6: Judgement scale to assess support for a feature.

Generic scale point	Definition of Scale point	Scale Point Mapping
Makes things worse	Cause Confusion. The way the feature is represented makes difficult its modelling and/or encourage its incorrect use	-1
No support	Fails to recognise it. The approach are not able to model a certain feature	0
Little support	The feature is supported indirectly, for example by the use of other model/approach in a non-standard combination	1
Some support	The feature is explicitly in the feature list of the model. However, some aspects of feature use are not catered for.	2
Strong support	The feature is explicitly in the feature list of the model. All aspects of the feature are covered but its use depends on the expertise of the user	3
Very strong support	The feature is explicitly in the feature list of the model. All aspects of the feature are covered and the approach provides a guide to assist the user	4
Full support	The feature appears explicitly in the feature list of the model. All its aspects are covered and the approach provides a methodology to assist the user	5

Table 7: Evaluation for the NFR Framework.

Feature	Imp	CAT	CSO	Dif	Sco
FR and NFR Representation	4	5	3	-2	-8
Collaborative Systems Characteristics	4	4	1	-3	-12
Awareness Representation	4	4	4	0	0
Quality Factors Representation	3	3	5	2	6
Importance of Requirements	3	3	0	-3	-9
Traceability	3	3	3	0	0
Quantitative Model	2	2	1	-1	-2
Hierarchical Representation	2	2	3	1	2
Model Complexity	1	1	3	2	2
Total					-21

Table 8: Evaluation for the *i** Framework.

Feature	Imp	CAT	CSO	Dif	Sco
FR and NFR Representation	4	5	5	0	0
Collaborative Systems Characteristics	4	4	5	1	4
Awareness Representation	4	4	5	1	4
Quality Factors Representation	3	3	5	2	6
Importance of Requirements	3	3	0	-3	-9
Traceability	3	3	3	0	0
Quantitative Model	2	2	1	-1	-2
Hierarchical Representation	2	2	3	1	2
Model Complexity	1	1	1	0	0
Total					5

Next, we should highlight that a variation of the DESMET method was used. 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 computation is shown in the column *Score* (Sco) in Tables 7, 8 and 9. 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 all the different GO approaches studied.

Figure 11 shows graphically the scores obtained by each one of the GO approaches. As can be observed, the *i** approach is the only one that has a positive score. Despite this positive score, it has been negatively evaluated for the Quantitative Model feature, since *i** only provides a partial support for quantifying the relations among requirements when using contribution links. The *i** approach also fails in representing the requirements importance, giving no support to determine which requirements are more important than others. Nevertheless, the other two GO approaches also share this lack of representation of the importance of each requirement. KAOS also fails in the same features than *i** but, unlike this approach, KAOS obtains a lower (or the same) score in almost all features except for the Hierarchical Representation feature, thanks to its tree-based representation. Finally, the NFR framework is the less suitable approach, obtaining a very low score, because of both the lack of expressiveness to specify FRs and its lack of adaptability to represent Collaborative Systems Characteristics.

Table 9: Evaluation for KAOS Methodology.

Feature	Imp	CAT	CSO	Dif	Sco
FR and NFR Representation	4	5	5	0	0
Collaborative Systems Characteristics	4	4	4	0	0
Awareness Representation	4	4	4	0	0
Quality Factors Representation	3	3	4	1	3
Importance of Requirements	3	3	0	-3	-9
Traceability	3	3	4	1	3
Quantitative Model	2	2	0	-2	-4
Hierarchical Representation	2	2	4	2	4
Model Complexity	1	1	2	1	1
Total					-2

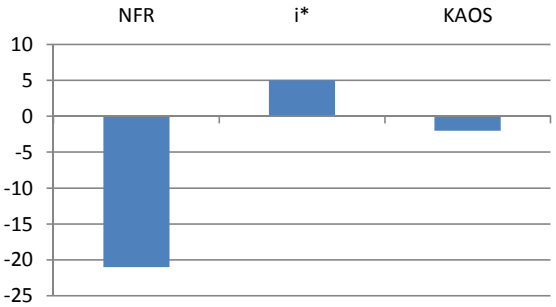

Figure 11: Empirical analysis results.

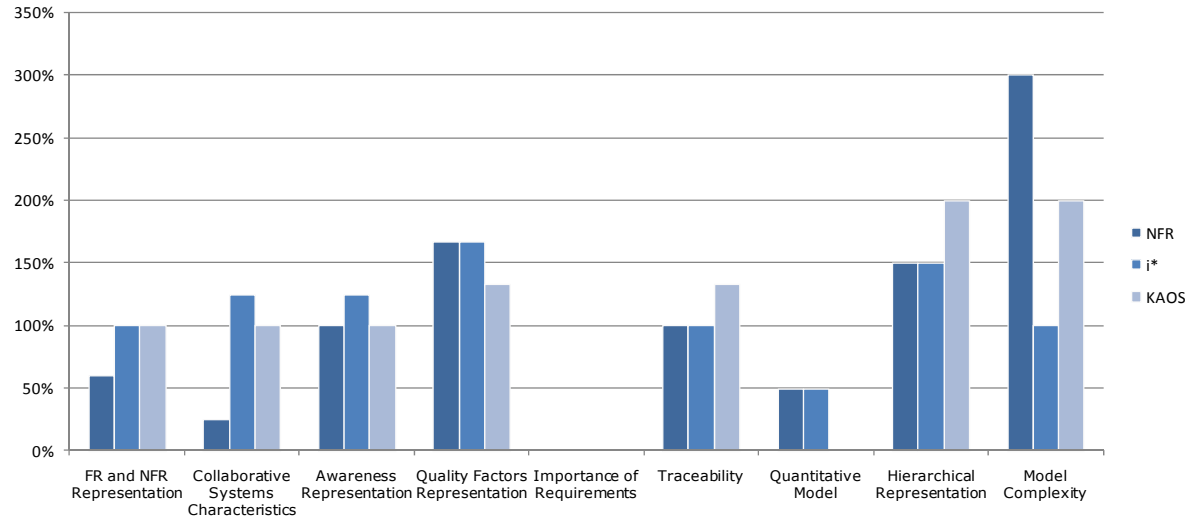

Figure 12: Results relative to distinct features.

In addition, as DESMET suggests, we have performed a comparative of the percentage of each feature satisfied by each analyzed GO approach. Figure 12 illustrates that the NFR approach only exceeds its competitors in the Model Complexity feature, due to the simplicity their models have. Similarly, KAOS supersedes *i** in this feature because *i** has more modelling elements for the sake of expressiveness. In addition, the evaluation of Hierarchical Representation and Traceability features for KAOS is better than for *i** because *i** models are usually defined following a network structure and do not provide a sophisticated support for traceability. Other meaningful fact is that no approach is able to represent the importance of the requirements, something that should be considered in future works. Another significant result is that, despite *i** and KAOS have the same score for the feature FR and NFR Representation, *i** supersedes KAOS in the most important features (mandatory and high desirable ones) except for the Traceability feature. Nevertheless, KAOS obtains a better score in the less valuated features, like Hierarchical Representation and Model Complexity.

6 CONCLUSIONS AND FURTHER WORK

Collaborative systems are highly demanding in terms of NFRs. Therefore, the selection of a RE technique with proper support for their successful specification is a must. In this sense, the exploitation of the GO approach emerges as the most appropriate proposal(Teruel et al., 2011). However, up-to-date several RE techniques have been proposed that follow this approach. In order to select the most suitable one, in this paper the results of an empirical experiment of several GO techniques considering the special needs of CSCW systems have been conducted.

After this empirical experiment, we can conclude that the analyzed GO approaches are not fully appropriate to model collaborative system characteristics and its relationships with awareness and quality requirements. Among the analyzed GO techniques, the *i** approach is the only one that has a positive score for the analyzed features related to CSCW systems. In addition, *i** is the only one that provides (partial) support for quantifying the relations among requirements when using contribution links. However, this technique also exhibits some shortcomings, such as the lack of a hierarchical representation or support for specifying the importance of the requirements. But perhaps, the most significant shortcoming is that the comprehensibility of the awareness requirements is not appropriate. For instance, *i** does not provide support to specify when a task is carried out by several roles, what is very common in a CSCW system.

These conclusions, along with the results shown in (Teruel et al., 2011), support our initial hypothesis: current Requirement Engineering techniques should be enriched to address the issues identified during this study regarding CSCW systems. As was shown in this study, *i** is the most promising technique to be used as the foundation for this improvement. This constitutes one of our future and challenging works: to adapt/extend *i**for this

kind of systems. In addition to this definition, its validation by means of more complex case studies is planned in the near future.

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

ACKNOWLEDGEMENTS

This work has been partially supported by a grant (DESACO, 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

Castro, J., Kolp, M. and Mylopoulos, J. (2001) 'A requirements-driven development methodology', 108-123.

Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. (2000) *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing.

Cockburn, A. (2000) *Writting Effective Use Cases*, Addison-Wesley.

Cysneiros, L. M. and Yu, E. (2003) 'Non-Functional Requirements Elicitation', in Sampaio do Prado Leite, J. C. and Doorn, J.H. (ed.) *Perspectives on Software Requirements*, Springer.

Cysneiros, L. M. and Yu, E. (2003) *Non-Functional Requirements Elicitation (Perspectives on Software Requirements)*, Springer.

Google (2011) *Google Docs*.

Gutwin, C. and Greenberg, S. (2002) 'A Descriptive Framework of Workspace Awareness for Real-Time Groupware', *Computer Supported Cooperative Work*, vol. 11, pp. 411-446.

Gutwin, C., Greenberg, S. and Roseman, M. (1996) 'Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation', 281-298.

Hochmuller, H. (1999) 'Towards the Proper Integration of Extra-Functional Requirements', *Australasian Journal of Information Systems*, vol. 6, no. 2.

Kavakli, E. and Loucopoulos, P. (2004) 'Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods', 102-124.

Kitchenham, B. (1993) 'DESMET: A methodology for evaluating software engineering methods and tools', in Rombach, H., Basili, V. and Selby, R. (ed.) *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, Springer Berlin / Heidelberg.

Moreira, A. M. D., Araújo, J. and Rashid, A. (2005) 'A Concern-Oriented Requirements Engineering Model', 293-308.

Mylopoulos, J., Castro, J. and Kolp, M. (2000) 'Tropos: A Framework for Requirements-Driven Software Development', 261-273.

Pohl, K. (2010) *Requirements Engeneering: Fundamentals, Principles, and Techniques*, Springer.

Sampaio, J.C. and Franco, A.P.M. (1993) 'A Strategy for Conceptual Model Acquisition', 243-246.

Schümmer, T. and Lukosch, S. (2007) *Patterns for Computer-Mediated Interaction*, John Wiley \& Sons Ltd.

Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) Quality model (2008).

Teruel, M. A., Navarro, E., López-Jaquero, V., Montero, F. and González, P. (2011) 'An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems', 15th Empirical Assesment of Software Engineering, Durham, UK.

van Lamsweerde, A. (2001) 'Goal-Oriented Requirements Engineering: A Guided Tour', Proceedings 5th IEEE International Symposium on RE, Toronto, 249-263.