Development of CSCW Interfaces from a User-Centered Viewpoint: Extending the TOUCHE Process Model through Defeasible Argumentation

María Paula González[1,2,4], Victor M.R. Penichet[3], Guillermo R. Simari[2], and Ricardo Tesoriero[3]

[1] National Council of Scientific and Technical Research (CONICET), Argentina
[2] Department of Computer Science and Engineering, Universidad Nacional del Sur
Av Alem 1253, 8000 Bahía Blanca, Argentina
[3] Computer Systems Department, Universidad Castilla-La Mancha
02071 Albacete, Spain
[4] GRIHO Research Group, Universitat de Lleida
25001 Lleida, Spain
{mpg,grs}@cs.uns.edu.ar, victor.penichet@uclm.es, ricardo@dsi.uclm.es

Abstract. The *Task-Oriented and User-*Centered *Process Model for Developing Interfaces for Human-Computer-Human Environments* (TOUCHE) is aimed to build up user interfaces for groupware applications under a Human-Computer Interaction perspective. It includes a large set of well known formal models like Class Diagrams, Organizational Structure Diagrams, Task Diagrams, Collaboration Diagrams and Abstract Interaction Objects among others. Most of such models, however, suffer from a number of limitations when formalizing users' commonsense. Over the last few years, Argumentation Systems have been gaining importance in several areas of Artificial Intelligence, mainly as a vehicle for facilitating rationally justifiable decision making when handling incomplete and potentially inconsistent information. This paper sketches a Proof of Concept to show how defeasible argumentation techniques can be embedded within the TOUCHE. The final goal is to enhance the capability of development process models for CSCW systems by including a rule-based approach for efficient reasoning with incomplete and inconsistent information.

1 Introduction and Motivations

Nowadays Computer Supported Collaborative Work (CSCW) is a challenging research field focused on developing groupware applications. In particular, the Task-Oriented and User-Centered Process Model for Developing Interfaces for Human-Computer-Human Environments (TOUCHE) [1,2] was formally defined, aimed to clearly describe design decisions when deploying CSCW interfaces under a Human-Computer Interaction (HCI) perspective. However, most of TOUCHE models suffer from a number of limitations when formalizing User-Centered commonsense and qualitative reasoning. Indeed, TOUCHE models only accounts for describing design

M. Kurosu (Ed.): Human Centered Design, HCII 2009, LNCS 5619, pp. 955–964, 2009.

issues omitting the record of the associated decision making process. As a consequence, only results of the discussion between members of the development team are traceable. But being able to support and document this decision making process can enhance TOUCHE capabilities, especially those related to maintenance and scalability. In this settings, TOUCHE can rely on argumentation techniques to solve the above problem by means of a rational justified procedure.

This paper describes how the TOUCHE model can be empowered through argumentation, in which knowledge representation and inference are captured in terms of Defeasible Logic Programming, a general-purpose defeasible argumentation formalism. We show how argument-based reasoning can be integrated in a TOUCHE System Requirement Document in order to provide a qualitative perspective capable to support and document part of the associated decision making process automatically.

First, some related works are described. Then, the TOUCHE process model is briefly described. Section 4 introduces some key concepts of defeasible argumentation. Next, Section 5 sketches a Proof of Concept to show how defeasible argumentation can be used in TOUCHE to model incomplete and probably inconsistent information, focusing the Case Study in the two first steps of the model. Finally, Section 6 concludes.

2 Related Work

To our knowledge there is not similar proposal for integrating defeasible argumentation and TOUCHE as described in this paper. An alternative approach is proposed by Design Rationale. Even though some models for Design Rationale are based on the use of arguments [3] none of them include an embedded engine capable of carrying out the automatic computation of defeasible arguments as in DeLP. The use of defeasible argumentation during requirement elicitation is shown in [4, 5]. These papers present the argument-based Goal Argumentation Method for justifying modelling decisions in goal-oriented requirements engineering, while our approach is based on a task-oriented methodology. The integration of argumentation techniques using DeLP within CSCW systems was proposed in [6]. Another example that shows the possibility of using argumentation to enhance CSCW capabilities is presented in [7], where a tool for drawing arguments was introduced.

Finally, it must be remarked that recent research has led to some interesting results to model dialectical discussions and negotiation in CSCW scenarios. For example, [8] have proposed a mechanism to manage dialectical discussions when a group of people want to collaboratively define requirements in natural language. However, their proposal is based on constraint satisfaction techniques, and does not consider the use of argumentation

3 The TOUCHE Process Model

TOUCHE is a process model and a methodology for the development of user interfaces for groupware applications from the requirements gathering up to the implementation stage [1,2]. It includes four development stages, namely: Requirements Gathering, Analysis, Design, and Implementation. The first stage (based on [9]) gathers the requirements of the system to be developed. Novel requirement gathering templates which

include some metadata that are important for the specification of groupware applications have been developed, as well as the System Requirement Document DRS [1]. In this first stage those templates provides the information to describe the organizational structure of the users in the system, the different participant (groups, users, individuals, and agents), the system objectives, the functional and non-functional requirements, and information requirements. All this information is provided according to CSCW criteria where the user as a member of a group of users is the main aspect to take into account.

The Analysis stage [2] studies the problem domain. Roles and tasks are identified and described from a structural perspective using Class Diagrams and the Organizational Structure Diagram; and from a behavioural perspective by means of the Task Diagram or TD (using CTT notation [10]). Besides, the Co-interaction Diagrams or CDs are included to identify relationships among the actors of the system. The Design stage provides a way to present the information (visualization, entries, controls, etc.) to the final user. All the information gathered up to now is processed and translated to a software representation. Users' awareness should be considered in order to get a good CSCW design. Abstract Interaction Objects (AIOs) are used to design Abstract User Interfaces (AUIs). We use the UsiXML conceptual scheme [11]. The model is enriched with a new AIO and several facets which provide more expressiveness to represent CSCW systems. Finally, the Implementation stage deals with the generation of the UI from the previous AIOs. It is a reification process from every component to more concrete elements according to implementation and platform details. The Cameleon process is follow. Several specific CIOs for groupware applications are proposed. The traceability between the defined models and between the different stages is also considered. [2].

As it can be seen, several formal models have been included within the TOUCHE methodology. However, those models expressed only final decision of the development team omitting the possibility of dealing with incomplete and possible inconsistent information, especially those associated with the decision making process. These limitations turn out to be critical especially in the first and second stage of the TOUCHE model, as decisions made during these two stages strongly condition the final product characteristics. As we will see next, defeasible argumentation -a sound setting formalization to model incomplete and possible inconsistent information- can cope with the problem described below.

4 Defeasible Argumentation

Argumentation Systems (AS) are increasingly being considered for applications in developing software engineering tools, constituting an important component of multi-agent systems for negotiation, problem solving, and for the fusion of data and knowledge [12,13]. AS implement a dialectical reasoning process by determining when a proposition follows from certain assumptions, analyzing whether some of those assumptions can be disproved by other assumptions in our premises. AS typically refer to two kinds of knowledge: strict and defeasible knowledge. Strict knowledge (K_S) corresponds to the knowledge which is certain; typical elements in K_S are statements or undisputable facts about the world, or mathematical truths (e.g. implications of the form $(\forall x)P(x) \rightarrow Q(x)$). The strict knowledge is consistent, i.e. no contradictory

conclusions can be derived from it. On the other hand, defeasible knowledge (K_D) corresponds to that knowledge which is tentative, modelled through "rules with exceptions" (defeasible rules) of the form "if P then usually Q" (e.g., "if something is a bird, it usually flies"). Such rules model our incomplete knowledge about the world, as they can have exceptions (e.g., a penguin, a dead bird, etc.). Syntactically, a special symbol ($\Rightarrow$) is used to distinguish "defeasible" rules from logical implications.

Example: *For the sake of example, let us consider a well-known problem of non-monotonic reasoning in AI about the flying abilities of birds, recast in argumentative terms. Consider the following sentences:*

1. Birds usually fly.
2. Penguins usually do not fly.
3. Penguins are birds.

The first two sentences correspond to defeasible rules (rules which are subject to possible exceptions). The third sentence is a strict rule, where no exceptions are possible. Now, given the fact that Tweety is a penguin two different arguments can be constructed:

1. Argument A (based on rules 1 & 3): Tweety is a penguin. Penguins are birds. Birds usually fly. So Tweety flies.
2. Argument B (based on rule 2): Tweety is a penguin. Penguins usually do not fly. So Tweety does not fly.

AS allow the user to define a knowledge base $K = K_S \cup K_D$ involving strict and defeasible knowledge. An argument A for a claim c is basically some "tentative proof" (formally, a ground instance of a subset of K_D) for concluding c from $A \cup K_S$ [13]. Arguments must additionally satisfy the requirement of consistency (an argument cannot include contradictory propositions) and minimality (by not including repeated or unnecessary information). Conflicting arguments may emerge from K. Intuitively, an argument A attacks another argument B whenever both of them cannot be accepted at the same time, as that would lead to contradictory conclusions.

There exist two main kinds of possible "attacks" between arguments in AS: symmetric attack (arguments with opposite conclusions) and undercutting attack (an argument attacks some "subargument" in another argument). The notion of defeat comes then into play to decide which argument should be preferred. The criterion for defeat can be defined in many ways, being a partial order $\leq$ among arguments. Thus, for example, arguments can be preferred according to the source (e.g. when having arguments about weather, the argument of a meteorologist should be stronger than the argument of a layman). As a generic criterion, it is also common to prefer those arguments which are more direct or more informed. This is known as the specificity principle (see [13]). For example, in the particular situation of the previous Example two arguments arise that cannot be accepted simultaneously (as they reach contradictory conclusions). Note that argument *B* seems rationally preferable over argument *A*, as it is based on more *specific* information. This situation can easily become much more complex, as an argument may be defeated by a second argument, which in turn can be defeated by a third argument, *reinstating* the first one. Indeed, AS determine

when a given argument is considered as ultimately acceptable with respect to the available knowledge by means of a dialectical analysis, which takes the form of a tree-like structure called dialectical tree. The root of the tree is a given argument A supporting some claim and children nodes for the root are those defeaters B_1, B_2, ... B_k for A. The process is repeated recursively on every defeater B_i, until all possible arguments have been considered. Leaves are arguments without defeaters. Some additional restrictions apply (e.g., the same argument cannot be used twice in a path, as that would be fallacious and would lead to infinite paths).

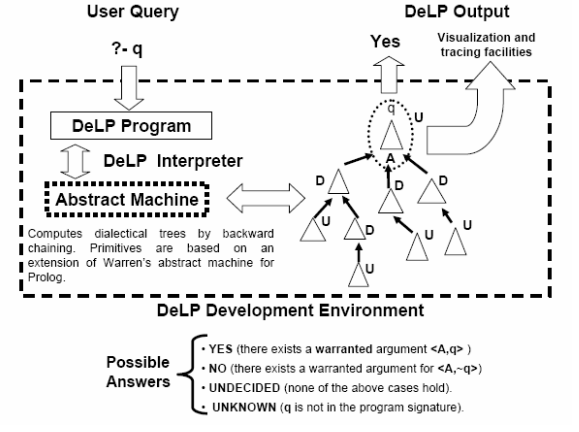

Fig. 1. A schematic view of the DeLP development environment

A marking procedure can be then performed for "marking" the nodes in the tree. Leaves will be "undefeated" nodes (or "U" nodes, for short), as they have no defeaters. Then we can propagate the marking from the leaves upward to the root as follows: an inner argument A_i in the tree will be marked as a "defeated" node ("D" node) if it has at least one "undefeated" child. Otherwise, if every child of A_i is a "D" node, then A_i will be marked as "U" node. If the root of a dialectical tree (the argument Arg) turns out to be marked as "U" node, then it is ultimately undefeated (given the knowledge available), so that the argument Arg (and its conclusion) is said to be warranted (i.e. ultimately accepted). Given a knowledge base K, AS automatically compute the dialectical tree associated with any particular claim (provided by the user as an input). In this context, Defeasible Logic Programming (DeLP)[1] is a general-purpose AS which has been particularly successful in real-world applications [14, 15, 16], providing an integrated environment for defining a knowledge base and solving user queries (claims) interactively. For any claim the DeLP engine automatically computes and visualizes the emerging dialectical tree, which acts as an explanation facility for the user, helping him to understand why the given claim is warranted or not (Figure 1). As we have seen in this Section, warranted arguments support beliefs that are accepted beyond dispute on the basis of the available knowledge. This notion can be applied in different contexts as, in particular, in multiple-party reasoning,

[1] See http://lidia.cs.uns.edu.ar/delp_client

where a group of several people participate on the basis of some common knowledge (e.g. members of a software development team). In the next section we will analyze how this idea can be integrated in the TOUCHE process model to characterize and document decision making by using of a general-purpose argumentation system like DeLP as a support tool.

5 Proof of Concept

5.1 Feasibility Analysis

First, our proposal requires an automated argumentation system to be integrated in TOUCHE, which should include an appropriate front-end for posing queries, and facilities for defining a knowledge base and visualizing results of the computation of the underlying argumentation engine (e.g. dialectical trees). Several of such kinds of platforms are freely available nowadays [16], covering the above expectations with reasonable costs (including economical resources, time consumption, etc.). Thus, AS (and particularly DeLP [16]) can be seen as a first step on the construction of argument-based modules to be completely embedded in the TOUCHE environment.

Costs associated with the inclusion of an extra theoretical framework in TOUCHE must be also considered. In that respect, note that existing argument-based platforms are not specially oriented towards experts on argumentation, but rather towards general users with a conceptual understanding about the meaning of facts, rules and inference by means of rule chaining. We claim that these concepts are suitable for TOUCHE users. The existence of graphical front-ends in some AS platforms [17] minimizes the complexity of text input for rules and facts as well as the interpretation of obtained results. Finally, from the TOUCHE viewpoint, it must be remarked that the integration of Artificial Intelligence and argument techniques within CSCW systems has proven to be fruitful, resulting in novel proposals [6,7,8] and systems such as I-MINDS [15] or SCALE [16] for example.

5.2 Case Study

The next example sketches how defeasible argumentation can be used in the first and second stages of the TOUCHE model to deal with inconsistent or incomplete information. Let us suppose we are developing the interface of a groupware application which allows several authors to create the same document through the Internet. When the authors of the document have written a draft, one of them is responsible for sending, through the same application, the document which is candidate to be published to some reviewers. Then, the reviewers discuss about the document and give their own opinion on whether it should be published or not. A published document can be read by all the users of the system, even if they are not authors or reviewers.

In the above settings, Table 1 shows part of the specification of a functional requirement called *document edition* (DC) by means of the metadata of the general template and the extensions introduced to consider the specific features concerning CSCW systems.

Table 1. Description of part of the functional requirement called *Document edition* with the proposed template (only relevant rows for the current case study are included)

RF-8	Document Edition
… … …	
Awareness issues	The following actors should be aware of this requirement: • #G-1 (AUTHORS): - *What*: an actor is modifying part of the current document - *How*: current modification is showed graphically - *When*: in real-time - *Where*: in the same workspace, in the same window - *Why*: to know who is modifying what and not to interfere - *What*: an actor modified a document - *How*: a past modification is showed by e-mail - *When*: after saving the current version, asynchronously - *Where*: in the actor's intranet and by e-mail - *Why*: to know who modified the document and what part
… … …	
CSCW description	Because of the collaborative nature of the current requirement: • Notifications are necessary for user awareness • Insertion, modification, and modification in a document are issues to be careful. Awareness in real time is important. Some actions such as deleting an image could be too fast for the rest of authors to be aware. They should be aware in some way. • Real-time feeling in the document elaboration is important but not vital.
… … …	

Note that some points of the Awareness Issue part of DC are in conflict with other consideration of the CSCW description. For example, the statements s_1= *"Awareness in real time is important"*, s_2= *"Some actions such as deleting an image could be too fast for the rest of authors to be aware"* are actually in conflict, while statements like s_3= *"Real-time feeling in the document elaboration is important but not vital"* are somehow vague or incomplete, and consequently their interpretation will be probably biased by the development team later on. At this point defeasible argumentation notation can be embedded into the template on Table 1 to enhance the comprehension of DC. For example, the following arguments can be associated with s_1, s_2 and s_3:

% Defeasible rules (Commonsense knowledge)
%W stands for an arbitrary writer, ┐ stands for "not" and T stands for "text"

– author (W) $\Rightarrow$ show_real_time (awareness, W)
– author (W), Deleting (W, T) $\Rightarrow$ show_real_time (awareness, W)
– author (W), Deleting (W,T), Image (T) $\Rightarrow$ ┐ show_real_time (awareness, W)
– coordinator (W) $\Rightarrow$ ┐ show_real_time (awareness, W)
– coordinator (W), author (W)$\Rightarrow$ show_real_time (awareness, W) %the coordinator is one of the authors

Later on, during the 2nd stage in TOUCHE, roles and tasks will be instantiated and modelled by means of some diagrams and descriptions (see Section 3). For example, DC will be designed by means of the framework shown in Figure 1. This time, arguments describing defeasible rules like those showed above can be used to decide how

to link elements in the diagrams of the second stage, especially in those cases where more than one solution must be taken into account. Here is when the potential of having an automated engine like DeLP, capable to compute arguments automatically comes into play. By means of the analysis of real Use Case Diagrams (see Figure 2 for the Use Case Diagram of the DC requirement) that can be easily collected and instantiated, members of the development teams can rely on AS like DeLP to compute alternative sets of facts automatically. This way, DeLP answers can be used to analyze alternative design responses for requirement descriptions minimizing the subjectiveness and the cultural bias present in the decision making process.

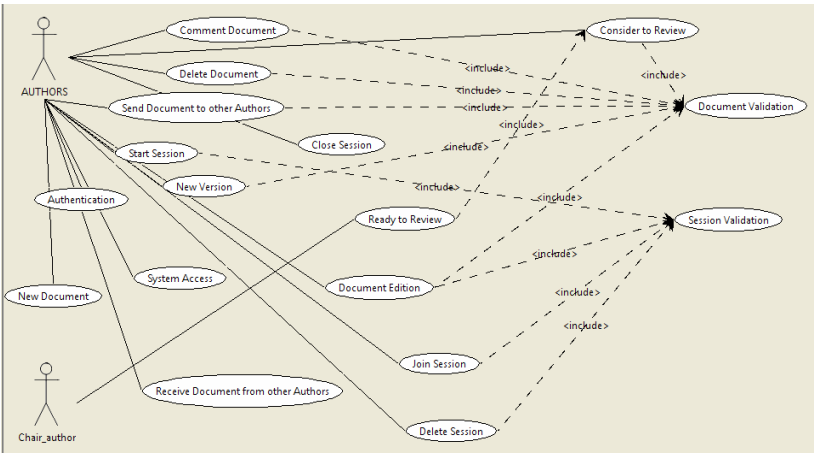

Fig. 2. Use Case Diagram for the requirement expressed in Table 1

6 Conclusions and Future Work

This paper proposed the integration of Argumentation Systems (AS) in the Task-Oriented and User-Centered Process Model for Developing Interfaces for Human-Computer-Human Environments (TOUCHE), aimed to build up CSCW interfaces from the Human-Computer Interaction viewpoint. The final goal is to enhance the capability of development process models for CSCW systems by including a rule-based approach for efficient reasoning with incomplete and inconsistent information. In our approach defeasible argumentation was captured in terms of Defeasible Logic Programming (DeLP), a general-purpose AS which has been particularly successful in real-world applications, providing an integrated environment for defining a knowledge base and solving user queries (claims) interactively. For any claim the DeLP engine automatically computes and visualizes the emerging dialectical tree, which acts as an explanation facility for the user, helping him to understand why the given claim is warranted or not.

As a first step in a novel research line, this work sketched a Proof of Concept to show how DeLP can enhance the TOUCHE first and second stages (namely Requirements Gathering and Analysis) respecting their capability to deal with incomplete, uncertain and possible contradictory information. Part of our future work includes the

development of a graphical module based on DeLP to be incorporated into the TOUCHE Case Tool.[2] Based on this incorporation, future work will focus on performing a set of experiments beyond this Proof of Concept. Indeed, it will be necessary to carry out a variety of complete CSCW interface developments under TOUCHE –comparing the results of including and omitting the use of DeLP– in order to validate the real scope of the current proposal. Work in this direction is being pursued.

Acknowledgments. We would like to acknowledge to CONICET (Argentina) and the projects PGI 24/N020, PGI 24/N023 (UNS, Argentina), PIP-CONICET 112-200801-02798 (CONICET, Argentina), CICYT TIN2008-06596-C02-01 (Spain) and the Junta de Comunidades de Castilla-La Mancha PAI06-0093-8836 for funding this work.

References

1. Penichet, V., Lozano, M., Gallud, J., Tesoriero, R.: Requirement Gathering Templates for Groupware Applications. In: Macías, J.A., Granollers, T., Latorre, P.M. (eds.) New Trends on HCI. Research, Development, New Tools and Methods. Springer, Heidelberg (2009)
2. Penichet, V., Lozano, M., Gallud, J., Tesoriero, R.: User Interface Analysis for Groupware Applications in the TOUCHE Process Model. International Journal Advances in Engineering Software (ADES) (in press, 2009) ISSN: 0965-9978, doi:10.1016/j.advengsoft.2009.01.026
3. Burge, J., Brown, D.C.: Reasoning with design rationale. In: AI in Design 2000, pp. 611–629. Kluwer Academic Publishers, Dordrecht (2000)
4. Jureta, J., Faulkner, S., Schobbens, P.: Clear justification of modelling decisions for goal-oriented requirements engineering. Requirements Eng. 13, 87–115 (2008)
5. Jureta, I.J., Faulkner, S.: Tracing the Rationale Behind UML Model Change Through Argumentation. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 454–469. Springer, Heidelberg (2007)
6. González, M.P., Chesñevar, C., Collazos, C., Simari, R.: Modelling Shared Knowledge and Shared Knowledge Awareness in CSCL Scenarios through Automated Argumentation Systems. In: Haake, J.M., Ochoa, S.F., Cechich, A. (eds.) CRIWG 2007. LNCS, vol. 4715, pp. 207–222. Springer, Heidelberg (2007)
7. Kirschner, P., Buckingham, S., Carr, C. (eds.): Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making. Springer, London (2003)
8. Gervasi, V., Zowghi, D.: Reasoning about inconsistencies in natural language requirements. ACM Trans. Softw. Eng. Methodol. 14(3), 277–330 (2005)
9. Durán, A.: Applying Requirements Engineering. Catedral Publicaciones, Spain (2003) ISBN: 84-96086-06-2
10. Paternò, F.: Model-based Design and Evaluation of Interactive Applications. Springer, Heidelberg (1999)
11. Limbourg, Q., et al.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In: Bastide, R., Palanque, P., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005)
12. Rahwan, I., Parsons, S., Reed, C. (eds.): Argumentation in Multi-Agent Systems. LNCS (LNAI), vol. 4946. Springer, Heidelberg (2008)
13. Chesñevar, C.I., Maguitman, A., Loui, R.: Logical Models of Argument. ACM Computing Surveys 32(4), 337–383 (2000)

[2] To access to the TOUCHE Case Tool consult www.penichet.net

14. Chesñevar, C., Maguitman, A., Simari, G.: Argument-Based Critics and Recommenders: A Qualitative Perspective on User Support Systems. Data & Knowledge Engineering 59(2), 293–319 (2006)
15. Brena, R., Aguirre, J., Chesñevar, C., Ramirez, E., Garrido, L.: Knowledge and Information Distribution Leveraged by Intelligent Agents. In: Knowledge and Information Systems (KAIS), vol. 12(2), pp. 203–227. Springer, Heidelberg (2007)
16. García, A., Simari, G.: Defeasible Logic Programming: An Argumentative Approach. Theory and Practice of Logic Programming 4(1), 95–138 (2004)
17. Reed, C., Rowe, G.: Araucaria: Software for Argument Analysis, Diagramming and Representation. Int. J. on Artificial Intelligence Tools 13(4), 961–979 (2004)
18. Liu, X., Zhang, X., Soh, L.-K., Al-Jaroodi, J., Jiang, H.: A Distributed, Multiagent Infrastructure for Real-Time, Virtual Classrooms. In: Proc. ICCE 2003, pp. 640–647 (2003)
19. Soller, A., Guizzardi, R., Molani, A., Perini, A.: SCALE: supporting community awareness, learning, and evolvement in an organizational learning environment. In: Proc. of the 6th international conference on Learning sciences, pp. 489–496 (2004)