A Method for Specification of Collaborative Interfaces through the Use of Scenarios

Anamaria Montandon Dumont
União de Escolas Superiores de Rondonópolis – FAIR/UNIR-MT
anamaria@unir-roo.br

Carlos Alberto Marques Pietrobon
Pontifícia Universidade Católica de Minas Gerais – PUC-MG
capietro@pucminas.br

Abstract

The complexity and the extension of tasks in the world today demand an extensive interaction among people with different knowledge backgrounds, who need to work in cooperation, aiming at a common goal. In this paper, we present a method to model the tasks and actions that a group of users carries out when working cooperatively through a computational system interface. For this purpose, we use scenarios for requirements gathering for user-cooperative interfaces. Along with the method, we propose the use of a notation that was developed to support interaction characteristics of users in cooperative work.

1. Introduction

Computational systems have been used to support completely different human activities, and they are essential work tools today. Recently, besides supporting the execution of individual tasks, they started becoming an important instrument of interaction and cooperation among people. However, this technology, even when used to support the activities carried out by a group, has been used to improve the individual's work, not the group's. For instance, text editors available enhanced the productivity of a person who is writing, but did not permit that two or more people write a document concurrently.

With the demand for softwares that support cooperative work, (Virtual Class, software development, etc), the challenge of specifying, designing and implementing these systems appears. One of the limiting points of these computational systems is the lack of an interface that supports appropriately groups of people that exerts its activities at the same time on the same representation or task. It is not properly an implementation problem, but rather a design one: there is a lack of method (and notation) means to express the dynamics related to using this kind of interface.

In this paper we present a method to obtain the requirements that a cooperative interface should comply with in order to support people working in a group. This method is based on the analysis of scenarios obtained from text descriptions of the tasks sent by the user.

Initially, we will approach the concept of cooperative work, the requirements of cooperative user interfaces and scenarios. Next, we will consider papers related to this area. In the following sections, we will deal with the method for obtaining cooperative interaction requirements from scenarios.

1.1. Computer-Supported Cooperative Work

In [15], Computer-Supported Cooperative Work (CSCW) "is considered a generic term that combines the understanding of the ways in which people work in groups with the available technologies (groupware) of software, hardware, services, and related techniques".

It's a research area that aims at integrating the works of several people involved in a common goal, inside a cooperative universe, through the sharing of resources in an efficient way, establishing, in this way, dynamic social and technological behaviors [24].

The requirements to support cooperative environments are [8], [15], [20], [24]: people coordination, information sharing (access concurrent control), workspace sharing, joint-work environment technical and social analysis, support to individual and joint work aspects, joint work dynamic control according to immediate necessities; group organizational memory, group dynamics, communication support, interaction transparence support, access policy establishment, recognition and dissemination of modifications.

From what was depicted above, it follows that the interface of collaborative systems is one of the vital aspects for the usability of such systems, for without it there is no communication, coordination or collaboration. And, exactly for these reasons, the design of such interfaces isn't an easy activity because it involves interaction among people.

2. Requirements Cooperative Interfaces

In this section it's presented a set of requirements that shall be complied with when people work cooperatively. These requirements will be used by designer and in the

user-specified cooperative activities extraction algorithm by means of scenarios, that will be showed in section 2.

Defining U = {u1, u2, ..., un} the set of system users in a cooperative activity. Defining E = {e1, e2, ..., en} the set of workstations where the several users are working in. For simplicity sake, let's also consider that for each user ui there exists one and only one workstation ej associated. Defining A = {a1, a2, ..., an} the set of actions that can be carried out by the user.

Defining ai an action carried out by the user ui that drives the computational system and for which a response is to be given (feedback). This response must be observed locally and in all other ek-1 workstations/users where it's sensible to have feedback for that action. This response can be produced by either a user or the computational system.

If we are considering a group of users, it's necessary not only to model the feedback, but also who will receive this feedback. Thus it's necessary to model the following relationships among the several users:

a) relationships 1-1 (one user communicating with other user)
b) relationships 1-n (one user communicating with several users)
c) relationships n-m (several users communicating with several users)
d) relationships n-1 (several users communicating with one user)

Other aspect that has to be distinguished is the feedback origin: system feedback or users feedback.

When dealing with groups, it's necessary to model the synchronism or asynchronism among actions and feedbacks, that is identify if two or more actions / feedbacks may/must happen at the same time or at different times.

3. Scenarios and the Use of Scenarios in the Software Development Process

According to [18], scenarios are "narrative descriptions of how an application can be used and include the context in which they occur, its goal, and a narrative of the sequence of executed actions". They may be produced by users and designers alike, with different purposes. When produced by the user, each user specifies the scenarios that describe his tasks. The designer can make use of the tasks identified to help the users identify their scenarios.

In [19], it's seen the use of scenarios in the different stages of the software development process. In [9], scenarios are used as a technique of elicit the application requirements, through the identification of actors, entities and actions that describe it.

In [18], is presented a method for gathering requirements and specification of interaction in a process of Software Engineering. The method consists in obtain use cases from scenarios and user interaction diagrams (a notation graphical proposed) from this method. These diagrams represent the interaction between the user and the application. This method, however, just does worry about obtaining of data and process; does not worry about approach as people interact cooperatively.

[21] presents an approach of HCI design oriented to web sites which is based on scenarios which describe interface aspects. In [9], scenarios are used to elicit the application requirements from the users and to refine and validate these requirements. [12] proposes an approach based on user stories and use cases for gathering requirements.

4. Proposed Method

In [18], the authors describe a method for requirements gathering and conceptual design. The requirements gathering stage presents the following phases: identification of actors and tasks, specification of scenarios, specification of use cases from scenarios, specification of user interaction diagrams (a graphical representation for the use cases), validation of use cases and user interaction diagrams. The following stage (conceptual design) has only one phase: specification of the conceptual scheme (a model of the application domain).

That method approaches only the exchange of information between the system and the user. In this work, we propose to model the interaction among the users starting from the "Specification of Scenarios" phase.

In this way, we intend to create the phase of "Specification of cases of collaborative interaction among the users", as an alternative way to specify aspects of collaborative interaction for multi-user applications. This phase is considered the key for the success of the collaborative applications development. The result of this phase is the "User Cooperation Diagram", which, graphically, represents the cooperation among people manifested in the scenarios. These diagrams are built using an existing graphic notation, which are the Sequence Diagrams (section 3), that specific the requirements. See Figure 1.

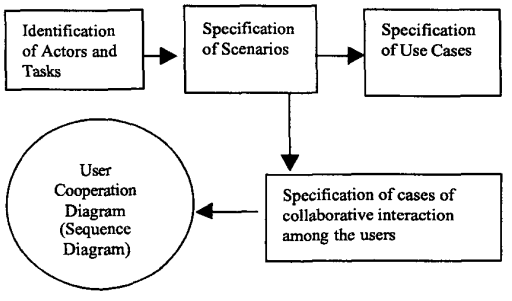

Figure 1. Phases of the requirements gathering of the proposed method.

To extract the cooperative activities from the scenarios, we propose the following steps (algorithm) to model or specify the collaborative interaction cases among the users. They are:

Identify the actors (1)
  Identify the users (2)
  Identify the hierarchies among the users (manager, leader, group, member) (3)
  Identify others situations, in which users are associated among themselves, organized as a group. These users appear in the scenarios as a group of people (4)
Identify tasks (decomposition and sequence) (5)
  Carried out individually/isolatedly (6)
  Carried out in groups (7)
    Specify collaborative interactions (8)
      Identify interaction categories (relationships 1-1; 1-N; N-M) (9)
        Identify types of interaction (10)
        Model the synchronism or assynchronism among the actions and feedbacks (11)
Identify products (12)
Identify others approaches (13)

Actors are people who exchange information among them (line 1 of the algorithm). They represent an specific type of user (line 2 of the algorithm), being a coordinator of others users, a group leader or an isolated member (line 3 of the algorithm). These actors may be organized according to a pre-established hierarchy among them. For instance, a member may be submitted to a group leader, which may be submitted to a manager. In a hierarchy it's necessary to provide "powers" to the higher levels to make them perform some kind of control on their subordinates. When the actors are at the same level of the hierarchy, they may form a group (line 4 of the algorithm). So that, it's necessary to identify and name these groups.

It's important to say that different actors can participate at the same scenario. As well as, the same actor can participate at different scenarios.

There will be variations on the vocabulary domination depending on the context. The actors identification will happen through NOUNS that represent the people involved in the context. For instance, in a company we can identify the actors through the words: president, director, manager, employee, and so on. while in a classroom we can identify: teacher, student, classmate, group leader and so on.

The tasks of each actor must be identified (line 5 of the algorithm). This identification will happen through VERBS that specify pertinent activities to the context. For example, in a company we can identify tasks through the verbs: protocol, send, count and so on, while in a classroom we can identify the verbs: ask, discuss, evaluate and so on.

A task can be decomposed in small tasks. We have to identify this decomposition and the sequence that these tasks are done (line 5 of the algorithm). The decomposition of the tasks shows that many people work at the system and, therefore, they are supposed to integrate with each other or with a coordinator who controls and/or integrates all the parts. The sequence of the tasks may also shows another way of interaction (workflow).

It's necessary to provide a short description of the tasks to identify which ones are done individually by only one user (line 6 of the algorithm), and which ones are done by a group (line 7 of the algorithm). This identification will happen through the use of the verbs in the SINGULAR or in the PLURAL. For example, when the user says "I do", "I send", we realize that he does the task alone. When the user says "we do", "we send", we realize that he does the task in group.

It's important to remind the tasks which are done individually must not be discarded at the first time, because they may be part of a cooperative activity bigger, in a decomposing structure or sequence.

For the tasks done by a group (line 7 of the algorithm) we need to identify the collaborative interactions (line 8 of the algorithm), in other words, we need to consider every relevant aspect in the interactions among the members of this group.

One of these aspects is the identification of the interaction categories (line 9 of the algorithm). We can classify three types of relationships among these members: a) relationships 1-1, b) relationships 1-N, c) relationships N-M.

In the first case (relationships 1-1) we are looking for expressions that identify that a member of the group communicates with only one person. For example, if the sentence "...if there is any problem, I send a message to the coordinator..." is described in a scenario, we realize that there is a communication among the elements from the group (send a message). This interaction happens between two people (the member and the coordinator).

In the relationships 1-N we are looking for expressions that identify that a member of the group communicates with several members. For example, the sentence "...I send the test for the students..."

In the relationships N-M we are looking for expressions that identify that several members of the group communicates with several members. For example, the sentence "...we discuss the question in the group..."

There are several situations that permit to identify the interactions, like: safety aspects (messages/data public or private and access permission) and alert situations such as errors of communication and warnings.

It's necessary to identify types of interaction (line 10 of the algorithm) such as the exchange of message/data and the exchange of social signals or non-techniques.

An example of synchronism (line 11 of the algorithm) is "...the teacher send the test at the same time for all the students..." and an example of assynchronism is "...the students send the test for the teacher at different times...".

The identification of the products (line 12 of the algorithm) which complement, and, in many cases, overcomes the identification of the tasks and actors suggests collaborative activities and the actors interaction, from manipulated products. So that, we ask about how each product was made (in group or not), if the end is in it or if it is part of a "big" product and who is going to use it.

There are others approaches (line 13 of the algorithm) that represent several kinds of situations the people have to get in touch with each others. It's impossible to enumerate all these situations; however, we must be awake to their existence. A good designer must have a list of them, got from the experience resulted by the development of systems. The designer, using its sensibility, may also notice others more complex situations do not related in this paper.

With these activities, we bring the requirements gathering for the support of a group working cooperatively. To put in documents these interactions, we propose the User Cooperation Diagrams. This is a Sequence Diagram [28],[29] specialized for our propose. The activities described in [18] only deal with data (model of data); we need as well to treat with the interaction (communication or interaction model) and with the interface (interface model), seen in [2]. The detailed description of the diagrams can be seen in the master's dissertation [27].

In the next section, it is presented an example of the application of the algorithm through the use of scenarios.

3. Example

The figure 2 below shows a simple example of an "User cooperation diagram" (Sequence Diagram [28], [29]) of the scenario "Employee receives a report, makes the analysis and send its opinion to the manager". Next we will comment about this example showing how we gather the requirements to make the diagram based on the scenario and using the algorithm proposed on section 2. During the explanation we will mention the meaning of the symbols used on the diagram.

At the scenario, we identify two users (line 2 of the algorithm): manager and employee (represented by the retangulars at the top of the Sequence Diagram in the figure 2).

To represent "Employee receives a report...", we realize that the manager creates the report which is represented at the Sequence Diagram by the arrow named "To Create Report" and send it to an employee, represented by the arrow named "To Send Report".

Note that several employees (relationships 1-N) (line 9 of the algorithm) might be receiving this report at the same time (synchronism) (line 11 of the algorithm). If we would like to represent this situation, we would use the notation in [27].

To represent "...makes the analysis...", we realize that the employee makes the analysis individually (line 6 of the algorithm), represented in the Sequence Diagram by the self-delegation symbol (an arrow that starts and finishes at the object, named "To Analyse Report"). This analysis could be done by a group (line 7 of the algorithm).

Note that the time the employee has spent to make the analysis, may be placed like a parameter, in brackets, for example (2 minutes), to represent time aspects.

To represent "...send its opinion to the manager" (arrow named "To Send Opinion"), we realize that the opinion sent to the manager is private (safety aspect) and that the relationship between the employee and the manager is 1-1 (line 9 of the algorithm).

Other aspects which were not modelled in this diagram can be analysed such as:

- The manager may send commentaries to the employee or store questions/answers in a repository so that the user has access all the time (assyncronism);
- The repository may work like a group memory to store tasks versions.

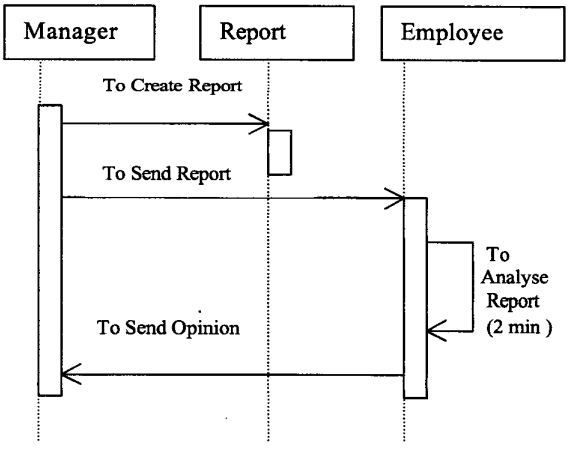

Figure 2. Scenario: "Employee receives a report, makes the analysis and send its opinion to the manager" and derived Sequence Diagram.

4. Conclusion

In this work, a method (a collection of activities) was proposed to specify collaborative interaction cases among the users from scenarios and using an existing notation (Sequence Diagram) to model these interactions.

There are several approaches of requirements gathering using scenarios. In general, these approaches use scenarios to determine data and functions modelled in different ways to which we have proposed in this paper. There is no knowledge of a method which uses scenarios to specify cooperation among users.

As we see, to support this group cooperative work, we need, among others aspects, to an interface adequated to this kind of work. The modelling of cooperative interfaces requires incremental methodologies which are not found in the literature. For this, we propose the method seen in section 2.

This work did not worry about implement aspects of the interface which the groups will support collaborating, but with the gathering and documentation of the requirements that have to be satisfied to support the activities of this group.

As well as scenarios, use cases can also provide information to this specification. In future works, use cases will be able to be used (explored) to provide more subsidy to reach an specification more completed. Collaboration diagrams will also be considered like an alternative documentation diagram.

We believe that our method and the notation that we use helps designers in the modelling of cooperative interactions. This method and notation can be seen much more detailed in the master's dissertation [27].

5. References

[1] Souza, C. S., Leite, J. C., Prates, R.O., and Barbosa, S. D. J., "Projeto de Interfaces de Usuário: Perspectivas Cognitivas e Semióticas"; National Congress of Society Brasilian of Computer, vol. II, 1999, pp. 425-476.

[2] Silva, E. J., "Projeto de Interfaces Homem-Máquina", 1999. IEC-PUC-MG.

[3] Maia, A.C., "A model for Cooperative Software Design"; 1996. http://www-nt.inf.puc-rio.br/cgilua/cgilua.exe/html

[4] Oliveira, O.L., "Interface extendida como um espaço de comunicação", 1999. IHC99.

http://www.unicamp.br/~ihc99/artigo_aceito.html

[5] Furtado, E., "Integrando fatores humanos no processo de desenvolvimento de interfaces homem-computador adaptativas", 1999, IHC99. http://www.unicamp.br/~ihc99/artigo_aceito.html

[6] Prates, R. O., and Souza, C. S., "Um modelo de apoio à expressão de projetistas de interfaces multi-usuário", 1999. IHC99. http://www.unicamp.br/~ihc99/artigo_aceito.html

[7] Barroso, G., "Teste de usabilidade de software" , 1998. Technical Report.

http://www-nt.inf.puc-rio.br/cgilua/cgilua.exe/html

[8] Grudin, J., "CSCW – History and Focus", 1994.

http://www.ics.uci.edu/~grudin/Papers/IEEE94/IEEEComplastsub.html

[9] Weidenhaupt, K., Klaus, P., Jarkem, M., and Haumer, P., "Scenarios in System Development : Current Practice" , IEEE Software , March/April, 1998.

[10] Bacelo, A. P., and Becker, K., "Uma ferramenta de apoio à discussão e deliberação em grupo", April, 1997. São Carlos – SP.

[11] Leite, J. C., and Souza, C. S., "Uma linguagem de especificação para Engenharia Semiótica de Interfaces de usuário", 1999. IHC99.

http://www.unicamp.br/~ihc99/artigo_aceito.html

[12] Shneiderman, B., "Designing the User Interface: Strategies for Effective Human-Computer Interaction", 3rd ed. , Addison-Wesley Longman, Reading Mass., 1998.

[13] Valaer, L. A., "Choosing a User Interface Development Tool", IEEE Software, July/August 1997.

[14] Pereira, A. C., and Sica, F.C., "Modelo para Construção de Serviços Cooperativos Baseado na Visão Vygotskyana e Formalizado pelos Conceitos Estruturalistas" , Technical Report (001/1999) – January, 1999 – Ouro Preto – MG.

[15] Pietrobon, C.A.M., Dumont, A.M., and Netto, O.A.M., "Extensão da Linguagem UAN para Especificação de Interfaces Cooperativas", Article, May, 2000.

[16] Souza, C. S., Prates, R. O., and Varejão, F. M., "Multimodal Communication between Software Designers and Software Users", Anais do III WorkshopHCI, May, 1997.

[17] Maia, A.C. P., Fuks, H., and Lucena, C. J. P., "A model of the communicative Behavior of Designers Involved in Cooperative Software Design", PUC-Rio, 1995.

[18] Vilain, P., Schwabe, D., and Souza, C. S., "Use Cases and Scenarios in the Conceptual Design of Web Applications", PUC-Rio, 2000.

[19] Breitman, K. K., and Leite, J. C. S. P., "Processo de Software Baseado em Cenários", PUC-Rio, 1999.

[20] Bock, G. E. and Marca, D. A., "Designing Groupware", August, 1995.

[21] Prates, R. O., Souza, C. S., "On the Rationale of Interface Semiotics for Multi-User Applications", 1999.

http://www.unicamp.br/~ihc99/artigo_aceito.html

[22] Christiansen, E., and Dirckinck-Holmfeld, L., "Making Distance Learning Collaborative", 1995.

http://www-cscl95.indiana.edu/cscl95/christia.html

[23] Collings, P., and Walker, D., "Applications to Support Student Group Work", 1995.

http://www-cscl95.indiana.edu/cscl95/collings.html

[24] Sica, F. C., "Camada Groupware: Uma camada de Suporte a Aplicações Cooperativas em Ambientes Distribuídos Abertos", Master's dissertation, 1996, Campinas – SP.

[25] Yourdon, E., "Análise Moderna Estruturada", 1989.

[26] Erikson, H.; Penker, M. – "UML Toolkit", September, 1997.

[27] Dumont, A. M., "Utilização de Cenários para Especificação de Interfaces Colaborativas", Master's dissertation, 2001, Pontifícia Universidade Católica de Minas Gerais – PUC-MG Brazil, Eletrical Engineering Departament.

[28] Fowler, M.; Scott, K. – "UML Distilled – Applying the Standard Object Modeling Language", Addison-Wesley, 1997.

[29] Harmon, P.; Watson, M. – "Understanding UML - The Developer's Guide", Morgan Kaufmann Publishers, 1997.