A methodological approach for user interface development of collaborative applications: A case study

Ana Isabel Molina*, Miguel Ángel Redondo, Manuel Ortega

Department of Information Systems and Technologies, School of Computer Science Engineering, University of Castilla – La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

**ARTICLE INFO**

Article history:
Received 10 July 2007
Received in revised form 29 December 2008
Accepted 2 March 2009
Available online 24 March 2009

Keywords:
Groupware design
Interaction design
Model-based design
Graphic User Interface design

**ABSTRACT**

In the last few years, the production of systems which support learning and group work has been high. However, the design and development of these types of systems are difficult, mainly due to the multidisciplinarity involved. Furthermore, the Graphic User Interface (GUI) of an application is receiving greater attention, since it can be decisive in determining if the application is accepted or rejected by users. Model-based design is a widespread technique in the user interface development process. While reviewing approaches that deal with the modeling and design of user interfaces supporting collaborative tasks, we have detected that there is no proposal that links interactive and collaborative issues. We have introduced a methodological approach to solve this shortcoming. This approach is called CIAM (Collaborative Interactive Application Methodology) and it is composed of several stages in which conceptual models are created using CIAN (Collaborative Interactive Application Notation). These models start by modeling the organization in which the application will be used, as well as the tasks that must be supported. In the initial stages, the organization and the collaborative tasks are modeled using high-level specifications. In the following stages, the level of detail increases and, finally, the interaction between the individual users and the application is modeled using ConcurTaskTrees (CTT) notation. The interaction model acts as a bridge between the design and the implementation of the Graphic User Interface. In this paper we present our methodological approach and an example of applying this method for user interface design of collaborative and interactive applications.

1. Introduction

The development of applications to electronically support the realization of group activities is a difficult task due to, among other things, the multiple disciplines that converge in their design process [14]. In order to facilitate this task, these applications can be framed in the paradigms of Computer Supported Cooperative Work (CSCW) or Computer Supported Collaborative Learning (CSCL) according to their specific purpose (group work or learning in-group). The difficulty comes from problems that can be placed in the following three dimensions: social nature, distributed systems and software engineering. These problems are related to aspects such as the support of cooperative and collaborative behavior modeling, the use of shared spaces and communication technology. However, these issues become fundamental requirements to consider during the development of these kinds of systems. The design of the interaction and the User Interface (UI) are particularly affected by these aspects.

* Corresponding author. Tel.: +34 926 295 300, fax: +34 926 295 354.
E-mail addresses: Anaisabel.Molina@uclm.es (A.I. Molina), Miguel.Redondo@uclm.es (M.A. Redondo), Manuel.Ortega@uclm.es (M. Ortega).

© 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2009.03.001
By reviewing Software Engineering and Human–Computer Interaction literature, we can observe the existence of some notations proposed for conceptual modeling of group work issues (see Section 2). Depending on their approaches, the proposals can be classified into the following categories:

- **User-centered notations**: to perform task analysis and modeling and based on techniques such as task decomposition (tasks/subtasks), specification of task flow (order), modeling of the objects associated with the tasks or information passing. Some examples are GOMS [5], HTA [3] and CTT [25].
- **Group-centered notations**: these explicitly approach the interaction between users who use a distributed application to carry out activities jointly.
- **Process-centered notations**: for modeling the processes that take place inside the organization, which is considered as a workflow with coordination necessities. In these models the aspects specified are relative to user behavior, data flow and interaction among agents (humans or computers).

The study of these proposals has allowed us to detect the following limitations [19]:

- The need for theoretical and computational models that allow the appropriate specification of computer-supported group activities.
- There are no notations that jointly approach interactive and group work issues.

None of the approaches studied supports the distinction between collaboration and cooperation. In many contexts, these terms are often used as synonyms. Dillenbourg [9] clarifies the difference, subtle yet important, between these two concepts:

- **Cooperation** entails the division of work to be performed, so that each person is responsible for his or her own portion of work. Members of the group pursue the same goals, but act independently in their own tasks, or perform the same task but in separate parts of the shared context.
- **Collaboration** entails the mutual commitment of the participants, as well as a coordinated effort to solve a problem. Collaboration is, therefore, a superior activity in which, in addition to cooperating, the members of the team have to work together on common tasks and towards a common outcome. The obtained result moves through different states to reach a state of final results obtained by the group. In the final product, it is difficult to determine the contribution of each member of the group. The collaboration assumes that the various members work within an area of common representation.

To take this distinction into account is interesting for us, as cooperation and collaboration imply different ways of understanding the division of tasks (which affects task modeling), the participation of the different roles in the development of these tasks (which affects the task and role modeling) and the product obtained as a result of this joint activity (which affects the data model). Furthermore, cooperation involves the inclusion of special coordination tasks at the end of the cooperative activity to enable the group to collect their individual contributions in the final product (group solution), as well as decision-making or agreements in this production process. In this latter case, we are talking about the existence of protocols for interaction and coordination among the group members.

This situation makes us raise the hypothesis of defining a notation to express collaborative and cooperative tasks in a differentiated way. This notation must make its characterization evident.

All these limitations cause the semantics of the specifications of collaborative applications to be incomplete. These problems bring to light the lack of a methodological framework that supports the design of the presentation layer of collaborative applications. In this paper our methodological proposal is presented. This approach is based on the use of specific notations for the design of interactive work group applications. This methodology connects high-level requirement models to low-level interaction models. The main focus of this work is on user interface design, but the models obtained as result of this methodology can be used as input in the existing approaches for deriving a concrete and final user interface. In Section 1, a review of the main contributions in this field is made, indicating each one's strengths and weaknesses. In Section 2, our methodological proposal is shown, enumerating its several stages and the aspects that are specified in each. Section 3 gives an example of its application, at the same time presenting notations used in each stage. Finally, the conclusions extracted from this work are presented, and the future work we plan to carry out is described.

### 2. Notations to specify group work issues

In this section, some of the main existing proposals for conceptual modeling of group work applications are presented. These approaches come from the Human–Computer Interaction (HCI) community, from Software Engineering (SE) and from the field of collaborative systems and, in particular, from workflow systems. Notations considered in this study are enumerated hereafter:

---

1 "Cooperation and collaboration do not differ in terms of whether or not the task is distributed, but by virtue of the way in which they are divided: in cooperation the task is split (hierarchically) into independent subtasks; in collaboration cognitive processes may be (heterarchically) divided into intertwined layers. In cooperation, coordination is only required when assembling partial results, while collaboration is <<...a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem>>."
The more relevant contributions in the field of HCI are:

- The ConcurTaskTrees (CTT) notation created by Fabio Paternó [25].
- The Group Task Analysis (GTA) Framework [39].
- The CUA (Collaborative Usability Analysis) notation [27,28].
- A task analysis method called MABTA (Multiple Aspect Based Task Analysis) [16,17].

In the fields of group work and workflow systems, we find:

- The Action Port Model (APM) notation [6,7], that is taken as reference by notations such as RML, TaskMODL and DiaMODL proposed by Traetteberg [36].
- The Proclets proposal [37,38].

As for approaches derived or completely framed within Software Engineering:

- The COMO-UML notation. UML does not include a suitable support for modeling collaborative activities. This has caused the definition of proposals for extending the standard notation. An example can be found in the COMO-UML notation. This notation is proposed for modeling of cooperative activities [12].
- The UML-G notation that makes use of extension mechanisms of UML for modeling cooperation requirements [31,32].
- The i∗ notation [40]. This notation allows a goal-oriented modeling and is used in disciplines related to requirements engineering or organizational processes modeling. There is a set of variants, among which we can highlight the Goal-oriented Requirement Language (GRL) [1] and TROPOS [2].

Each of these notations presents a different way of approaching group work systems modeling. We are interested in their conceptual and methodological issues; that is, whether or not these notations are set in a methodological framework, as well as whether or not they are supported by the definition of a conceptual framework (ontology) that allows the reduction of ambiguities and establishes a basis for the aforementioned notations. Also, we are interested in learning about their support for group work conceptual modeling, as well as pointing out whether these techniques distinguish between cooperation and collaboration in the terms outlined in [9].

In addition, we are interested in knowing which ones contemplate modeling interactive aspects (usually based on the distinction between user and application tasks) and, especially, if they tackle automatic or semiautomatic user interface generation or the derivation of some type of software architecture for supporting the application. We also analyze whether these proposals have a CASE tool for supporting them. Finally, for each notation we are going to point out its leading contribution that makes it stand out from the others (i.e. the contribution that we consider most remarkable with respect to the rest). The result of the analysis is shown in Tables 1 and 2.

In Tables 1 and 2 we present notations for the group work applications modeling that we consider most representative and we highlight their main characteristics. In Table 1, we present and summarize general and fundamental features: approach, ontology definition, the methodology in which they are framed and their most outstanding contribution. In Table 2, issues mainly related with modeling are summarized.

As a result of the comparative study of the main contributions in the field of group work modeling, we have reached the following conclusions:

- In the context of Human–Computer Interaction, usually modeling support is based on the extension of the existing notations by means of the incorporation of three new concepts:
  - The use of a new kind of task in the models (the cooperative task), which is also divided into individual tasks, carried out by the different roles involved in the system.
  - The possibility of indicating which roles perform which tasks.
  - The separation into models: cooperative and individual models. In the latter, an interactive task tree is created for each role involved in the group work.
- As for the model validation, GTA includes the use of ontology to validate the consistency between the various models proposed for specifying the system. Both CTT and COMO-UML allow the transformation of the models created using these notations into formal representations, such as LOTOS and Colored Petri Nets, respectively. Using formal notations allows the created models to be simulated and validated.
- With the exception of COMO-UML, the rest of the studied proposals present the following problems:
  - They presuppose static requirements, so the real behavior of the group is not correctly modeled. This behavior has a dynamic nature (actors that can play several roles at different times, etc.).
  - They do not have a good connection with methodologies from the Software Engineering field.
- As for the type of notation used for the task analysis and modeling, most proposals share the following characteristics that we consider requirements that must be taken into account in a proposal aiming to tackle the modeling of interactive and group work aspects:
  - Task decomposition. This is the usual way that human beings work: decomposing tasks into subtasks of lower difficulty. There are several methods based on this feature for structuring and managing the complexity of the specifications. The use of a representation in the tree form to express the decomposition of tasks into simpler tasks is common (examples...
Table 1
General and fundamental features of notations for group work applications modeling.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Approach</th>
<th>Ontology</th>
<th>Methodology</th>
<th>Leading contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTT</td>
<td>Group centered</td>
<td>No</td>
<td>No</td>
<td>- It is an improvement of a notation from the HCI field.</td>
</tr>
<tr>
<td></td>
<td>(Cooperative model)</td>
<td></td>
<td></td>
<td>- Distinguishing between Cooperative and Individual Model (tasks of a role).</td>
</tr>
<tr>
<td></td>
<td>- User centered</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(A model for each role)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTA</td>
<td>Group centered</td>
<td>Yes</td>
<td>Yes (DUTCH)</td>
<td>- Uses ontologies for model validation.</td>
</tr>
<tr>
<td></td>
<td>- User centered</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUA</td>
<td>Group centered</td>
<td>No</td>
<td>No</td>
<td>- Identifies a set of basic mechanics of collaboration.</td>
</tr>
<tr>
<td>MABTA</td>
<td>- Group centered</td>
<td>No</td>
<td>Yes</td>
<td>- There are several modeling levels, in which several contextual aspects are considered.</td>
</tr>
<tr>
<td></td>
<td>- User centered</td>
<td></td>
<td></td>
<td>- Task classification (coordination tasks, decision-making tasks, etc.)</td>
</tr>
<tr>
<td>APM</td>
<td>- Process centered</td>
<td>No</td>
<td>No</td>
<td>- Resources Modeling with interaction between users.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Interaction Modeling based on the speech acts theory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Using templates or construction blocks and patterns.</td>
</tr>
<tr>
<td>RML, TaskMODL &amp;</td>
<td>- Process centered</td>
<td>No</td>
<td>No</td>
<td>- It incorporates different languages for domain, tasks and dialog modeling.</td>
</tr>
<tr>
<td>Di-</td>
<td>- User centered</td>
<td></td>
<td></td>
<td>- Patterns.</td>
</tr>
<tr>
<td>aMODL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proclets</td>
<td>- Process centered</td>
<td>No</td>
<td>No</td>
<td>- Process Instantiation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Models reuse (components).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- It explicitly tackles interaction among proclets.</td>
</tr>
<tr>
<td>COMO-UML</td>
<td>- Group centered</td>
<td>Yes; Defining a Conceptual Framework</td>
<td>Yes (AMENITIES)</td>
<td>- Suitable integration with Software Engineering issues (notation derived from UML).</td>
</tr>
<tr>
<td></td>
<td>- User centered</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UML-G</td>
<td>- Group centered</td>
<td>No</td>
<td>No</td>
<td>- Clear identification of requirements to be considered in the modeling of groupware systems (shared data).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Use of extension mechanisms of UML for modeling these requirements.</td>
</tr>
<tr>
<td>Notation i* and</td>
<td>- Group centered</td>
<td>Yes</td>
<td>Yes (TROPOS is a methodology)</td>
<td>- It allows the clear and simple expression of the actors' goals and dependences among them.</td>
</tr>
<tr>
<td>derived (GRL,</td>
<td>(SD model)</td>
<td></td>
<td></td>
<td>- Differentiation between internal and external intentional elements.</td>
</tr>
<tr>
<td>TROPOS)</td>
<td>- User centered</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

can be found in notations such as HTA, CTT, the work modeling in GTA or the MABTA notation). This decomposition can be done in tree form or can be based on the creation of new models representing specifications of a lower level of abstraction, as is the case in APM and COMO-UML notations.

- Task Flow Specification. This is another aspect present in most of the specification techniques for specifying group work. The task flow allows the order of execution to be indicated. The order of the tasks is represented in most notations using temporal operators.

- Data Modeling. Although the modeling of objects is closest to the design and implementation of the final tool, a subset of these objects must be shown and/or manipulated by the user interface, as well as modified by the tasks performed by the application.

- To these initial characteristics, we have to add those that are suitable for specifying group work. In this regard, we point out the following:

- Modeling of the organization. The current modeling techniques must include user characteristics as a member of a group or organization. Their position within a group can be specified through their responsibilities (i.e. the role played), and within the organization by the position occupied in its hierarchy. Most existing notations do not provide the modeling of hierarchical relationships within a group. However, this kind of vertical relationship affects the work to be performed by an individual who may be assigned or delegated tasks. This dynamism may also be made horizontally between people who are at the same level in the hierarchy. The modeling of these dynamic aspects is very useful in specifying an organization that is as realistic as possible.
Table 2
Specific issues related to notations for group work applications modeling.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Interactive issues modeling</th>
<th>Group work issues modeling</th>
<th>Distinguish between collaboration/cooperation</th>
<th>Modeling supporting tool</th>
<th>Support for code automatic generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTT</td>
<td>Yes, distinguishing between interaction and application tasks, as well as temporal relationships among them.</td>
<td>Roles, Cooperative Tasks, Objects (not graphically expressed).</td>
<td>No (only cooperation modeling).</td>
<td>Yes (CTTE)</td>
<td>The application TERESA generates code for mono-user applications, but not for multiuser.</td>
</tr>
<tr>
<td>GTA</td>
<td>Yes, incorporating specifications in UAN notation.</td>
<td>Roles, Objects, Agents, Events, Tasks.</td>
<td>No (Only cooperation modeling).</td>
<td>Yes (EUTERPE)</td>
<td>No</td>
</tr>
<tr>
<td>CUA</td>
<td>Yes, individual tasks can be represented using HTA notation.</td>
<td>Scenarios, Tasks, Task instantiations (Individual and collaborative instantiation), Actions.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MABTA</td>
<td>Yes, in lower-level abstraction stages, the use of HTA notation is proposed.</td>
<td>Operation, Action, Activity, Aspect.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>APM</td>
<td>No</td>
<td>Processes, Actions, Resources (Actors, Tools or Information Objects), Ports, Flows.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RML, TaskMODL &amp; DiaMODL</td>
<td>Yes</td>
<td>Roles, Actors, Groups, Actions, Tasks, Objects, Tools, Events</td>
<td>No</td>
<td>Yes (Visio)</td>
<td>No, but transformation is almost immediate.</td>
</tr>
<tr>
<td>Proclets</td>
<td>No</td>
<td>Process (Proclets), Channels, Messages (performatives), Ports, Naming Service, Actors, Tasks.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>COMO-UML</td>
<td>No</td>
<td>Roles, Cooperative Tasks, Interaction Protocols.</td>
<td>No (Only cooperation modeling)</td>
<td>Yes (COMO-TOOL)</td>
<td>No</td>
</tr>
<tr>
<td>UML-G</td>
<td>No</td>
<td>Shared Data, Notifications, Actors, Roles, Shared Activities.</td>
<td>No</td>
<td>Any tool that supports UML modeling (Rational Rose, UML Argo, etc.).</td>
<td>No</td>
</tr>
<tr>
<td>Notation i* and derived (GRL, TROPOS)</td>
<td>No</td>
<td>Actors (Agents, Roles), Dependences, Goal, Task, Resource, Intentional Limits.</td>
<td>No</td>
<td>Yes (OME)</td>
<td>No</td>
</tr>
</tbody>
</table>

- **Differentiation between roles and actors.** Both terms represent different concepts whose definition must be made clear. A **role** is defined as the set of tasks or responsibilities to be played by one or more actors. An **actor** is a subject, human or not, that interacts with the system and can play a role at any given time. With regard to the actors, we are mainly interested in knowing their characteristics (for example, their skills in using computers, their specific skills in a particular area, their language, etc.). The actors do not necessarily need to be specific individuals; they may represent classes of individuals who share certain characteristics. The actors perform tasks but always in the form of a specific **role**.

- **The adoption of a specific role** by an actor can be due to assignment, delegation, mandate or any other situation involving the work context.

- The distinction between roles and actors implies the adoption of **user identification systems**, as well as **user interface adaptation** to support the tasks assigned to a particular role. Moreover, knowing the characteristics that define specific actors allows certain features of the system to be adapted to their preferences and/or skills.

- Some of the proposals dealing with the modeling of the dynamic aspects of a system add the concept of **event**, which may be triggered or take place without users having control over the initiation of these events.
• As is outlined in [15], conceptual models must contemplate *modeling of time constraints*. The temporal information that should be taken into account when analyzing and modeling user tasks can be classified into two categories:

  - **Qualitative temporal information.** Most of the notations consider a qualitative approach with regard to time modeling. In most cases this is limited to specifying a partial order among the activities to be carried out by the system. The specification of temporal relationships is based on temporal operators, such as sequence, iteration, choice, etc.

  - **Quantitative temporal information.** This kind of information is related to delays, dates and durations. There are few notations that explicitly include modeling of quantitative temporal information.

As for the notations analyzed, CTT includes both kinds of temporal information. This notation includes a wide set of operators that provide qualitative temporal information. Furthermore, this notation also allows the specification of certain quantitative temporal information, although it is not expressed graphically. The editor of CTT models (the CTTE tool) allows properties that are associated with the tasks, and related to their duration, to be included (minimum, maximum and average amount of time for the performance of a task). Nevertheless, there is other kind of quantitative temporal information that is not supported by this notation. For example, it is not possible to specify that at a given time a task terminates or to include temporal information (dates, time constraints) associated with flow between tasks.

With regard to these aspects it would be desirable to include both types of information in a proposal that addresses the modeling of group work. That is, a set of temporal operators for expressing order and, therefore, a way to specify the coordination between tasks (qualitative information) and also periods of time (quantitative information). In this second sense, we will consider the representation of periods of time, dates associated with flow between tasks and time constraints.

As a result of this comparative analysis we can conclude that although COMO-UML seems to be the most complete of all the studied proposals, this approach leaves some issues incomplete: collaboration and interactive issues. Considering the features identified during the study of these approaches, as well as what some of them lack, we consider it necessary to have a method and tools that guide software engineers in the modeling process of interactive issues in groupware applications. We propose a process that aims to provide a solution to those shortcomings, possessing the interaction and collaboration requirements as objectives to analyze in order to derive the user interface of the application to support them. In the next section, this methodology is introduced.

### 3. Methodological approach

In this section, we present the stages that comprise our methodological approach to deal with the conceptual design of interactive applications for supporting group work. Our proposal (CIAM, *Collaborative Interactive Application Methodology*) implies the adoption of different points of view when tackling the creation of the conceptual models of groupware applications. The objective of this approach is to serve as a guide for the designer when creating a conceptual specification of the main aspects that characterize the groupware systems and lead to the design of their software architecture and to the design of the interaction that these systems support. The information specified in each of the stages serves as a basis for the modeling to be carried out in the following stage, so that this information is extended, related or specified with a higher level of detail.

The first stages tackle *group-centered* modeling, going on in subsequent stages to a more *process-centered* modeling (*cooperation, collaboration* and *coordination*), approaching, as we go deeper into the abstraction level, a more *user-centered* modeling, in which interactive tasks are modeled (that is, the dialog that occurs between an individual user and the application). So, the first stages manage a high level of abstraction while, as they progressively advance, they go deeper into aspects that describe the interaction that takes place with a high level of detail. The two first stages allow the *definition of the context* in which the interactive model will be created, and thus, this serves as a starting point for the last ones. In this way, collaborative aspects (groups, processes) and interactive (individual) modeling are tackled jointly.

The stages in this proposal and their objectives are enumerated next. **Fig. 1** shows these stages and how they are related to each other.

- **Sociogram Development.** In this stage a diagram, called a sociogram, is created. This diagram allows the organizational structure to be represented, as well as the relationships between its members. Organization Members are in one of these categories: *roles, actors, software agents*; or groupings of the previous ones, giving rise to *groups*, that is, groups of actors with homogeneous responsibilities, or *work teams*, consisting of several roles. The elements in these diagrams might be interconnected by means of three kinds of basic relationships:
- **Inheritance** relationship. Responsibilities can be inherited whenever a certain precondition occurs (which can be added to the model).
- **Acting** relationship. This allows actors and roles to be related. This relationship can be annotated by cardinality (the number of actors that can play a certain role in the organization).
- **Association** relationship. This kind of relationship allows roles to be associated with each other, indicating that there are situations in which these roles cooperate or collaborate to carry out a joint task.

**Process modeling.** In this stage, the main collaboration/cooperation tasks, which define the group work developed in the organization previously defined, are described. A collaboration/cooperation process consists of a set of tasks carried out in a certain order and considering certain data or temporal restrictions among them. For each task, the roles involved, the data manipulated and the product obtained as a result of the task, are specified. For the data specified in the context of a task, we can specify the access modifiers to the objects, which can be reading, writing or creation. Each task must be classified in one of the following categories: cooperative task, collaborative task or individual task. The tasks in the process will be interconnected by means of several kinds of relationships that can be interpreted as dependences: temporal dependences (order relationship), data dependences (when tasks need data manipulated by previous tasks) and notification dependences (when it is necessary for a certain event to occur so that the workflow continues). In addition, to these types of relationships between tasks, it is necessary to consider dependences of an individual nature, that is, the iteration of a certain task or the maintenance of a certain condition during its execution. In this case, the following types of conditions are distinguished: completion condition (ending conditions for iterative execution), period condition (indicating temporal events for task completion) or execution condition (conditions indicating optional execution). In this stage, two descriptive techniques are used for specifying all this information. They are the Participation Table and the Process Model that will be presented in detail in the next section.

- **Responsibilities modeling.** Once group work has been specified at a high level by means of the Process Model, attention is paid to the individual perspective of each organizational member, that is, their roles. Using the information specified in the previous stage, the responsibilities of each role are extended, adding individual responsibilities that are not developed in group. These responsibilities imply carrying out tasks/ actions that handle objects. Therefore, for each task the object manipulated has to be specified, as well as the prerequisites that allow their correct execution (that is, the tasks/actions need to have been completed and the data created before a given task/action begins). The information detailed in this stage is supplementary to the one in the previous stage, being necessary for both models to be coherent (for example, when the present task needs to have some data, a previous task should have already created them).

- **Group work task modeling.** In this stage the group work tasks identified in the previous stage are described in greater detail. Considering the differentiation pointed out by Dillenbourg [9], we distinguish two different kinds of tasks, which need to be modeled in a differentiated way:
  - **Cooperative tasks.** They are modeled using a descriptive technique, similar to the one used for creating the process model, called graph of responsibilities decomposition. The subtasks that compose a cooperative task are detailed, so that, in the lower level of abstraction only individual tasks must appear. When specifying the data manipulated by a certain task, we can handle several levels (Object Collection/Objects/Attributes). Thus, there are situations in which different instances of the same object or collection of objects are manipulated, and others in which different pieces of the same information that comprise an object are used in the context of a certain task (even manipulation at an attribute level). This last case involves situations of concurrency of tasks. Also, different versions from the same object can be handled. This situation implies the existence of a temporal (non-concurrent) relation between the different tasks involved in the process of cooperative work.
  - **Collaborative tasks.** Collaborative Task modeling requires the specification of the roles involved in its execution, as well as the objects of the data model manipulated and shared by the work team. The proposed notation contemplates the specification of the shared context [11]. The shared context is defined as the set of objects that is visible to users, as well as the actions that can be executed on those objects. Once the objects that comprise the shared context have been decided, it is necessary to segment this information into three differentiated parts: objects and/or attributes manipulated in the collaborative visualization area, those that are manipulated in the individual visualization area, and those that comprise the segment of exclusive edition (a subset of the data model accessed exclusively by only one application user at a time).

- **Interaction modeling.** In this phase, the interactive aspects of the application are modeled. For each task of an individual nature detected in the previous stages of the process, an interaction model is created. The interaction is expressed by means of a decomposition tree of interactive tasks, using CTT notation [25]. As for the collaborative tasks, the interaction model is extracted directly from the shared context definition.

- **GUI design.** The user interface design stage is based on the interaction models obtained in the previous stage. This final stage, which we have labeled as MBUID (Model-Based User Interface Development), indicates that it is possible to apply, at the end of our process, any MBUID method or environment [29] existing in the literature, whenever it starts from a modeling of the interactive tasks. Model-based methods provide a mechanism to design the UI by means of a set of declarative models that describe not only the static and dynamic aspects of the user interface, but also all its relevant factors, which include the requirements related to the usage context. Then, these declarative models are translated into code, directly executable on a specific platform, or into some kind of intermediate language (usually an XML-based
Fig. 2. CIAN Notation Summary.

language), which can be interpreted by a renderer. In short, model-based methods suppose a UI design and development method, giving the required formalisms to build user interfaces systematically [30, 36, 4, 18].

The models created in each of the stages of the proposal are specified using a set of graphical elements that are summarized in Fig. 2. On the top left (2.a) of the figure, we can see the icons that represent the organization members (roles, actors, software agents, etc.). On the bottom left (2.a) and the top center (2.b) areas, we can see the icons for representing the nodes that form the Process Model and for indicating the different tasks and interdependence types. On the right-hand side (2.c), we can see the icons used for representing an interaction task model in CTT notation. We have enriched this notation by means of the use of three new icons to express visualization features and blockade of the objects that comprise the shared context in a collaborative task. A more detailed description of CIAN (Collaborative Interactive Application Notation) notation can be found in [21].

In the following section, and by means of an example, we will show how to use this notation and the models generated in each stage. In addition, we are going to present how interaction trees are extracted. In Table 3, we summarize the specification techniques used in the different stages of our methodological approach, the notation used and the product obtained in each of them.

4. Application example

In this section, a complete example of the application of our method for user interface development of collaborative applications is explained. For each of the stages previously presented, we will show the models obtained and we will explain elements that comprise the notation proposed for creating each one. We have chosen the “Conference Review System” as a case study, extracted from [34]. We have to point out that our interest focuses on user interface development.

Problem Definition
The purpose of the system is to support the process of submission, evaluation and selection of papers for a conference. The roles participating in this system are summarized in Table 4.

The following functions must be supported by the system:

- Any registered author may submit a paper. To do this, the author must inform the authors of the paper (all of whom must be registered) of the title, the abstract, the conference track, and a set of subjects chosen from a list previously determined by the PCChair, if there is one. The system, after checking the authors’ registrations, assigns a paper ID to the new paper, and allows the user to submit it by uploading a file. At any time, an author is allowed to check data about his submitted paper(s). Until the submission deadline, the author is also allowed to substitute the uploaded file with a new one, or to change any of the data that was provided for the paper.

- Once the submission deadline has been reached, the PCChair opens the list of submitted papers for perusal by the PCMembers, who may indicate their interest (an integer in the range 0–2, where 0 indicates no interest, 1 indicates moderate interest and 2 indicates strong interest). The PCChair, looking at the list of preferred papers of the PCMembers, as well as their indicated preferred topics and subjects, assigns papers to PCMembers for reviewing. Once this assignment
Table 3
Specification techniques and result obtained in the several stages of CIAM.

<table>
<thead>
<tr>
<th>CIAM methodology stage</th>
<th>Specification technique</th>
<th>Representation type</th>
<th>Result obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sociogram development</td>
<td>Sociogram</td>
<td>Graphic</td>
<td>Organizational structure specification, (roles, actors, software agents, etc.)</td>
</tr>
<tr>
<td>Responsibilities modeling</td>
<td>Responsibilities model</td>
<td>Textual</td>
<td>Detailed specification of the responsibilities of each role.</td>
</tr>
<tr>
<td>Process modeling</td>
<td>Participation table</td>
<td>Textual</td>
<td>Relationships between the main tasks and roles of the system.</td>
</tr>
<tr>
<td></td>
<td>Process model</td>
<td>Graphic</td>
<td>Work structure and workflow to be performed by the organization.</td>
</tr>
<tr>
<td>Group work task modeling</td>
<td>Access control matrix</td>
<td>Textual</td>
<td>Relationships between data objects and roles at group work task level.</td>
</tr>
<tr>
<td></td>
<td>Table of operation permissions on the shared context</td>
<td>Textual</td>
<td>Relationships between operations and roles at group work task level.</td>
</tr>
<tr>
<td>Collaborative task modeling</td>
<td>Responsibilities decomposition graph</td>
<td>Graphic</td>
<td>Responsibilities distribution in a cooperative task specification.</td>
</tr>
<tr>
<td>Cooperative task modeling</td>
<td>Shared context specification (class diagram in standard UML notation)</td>
<td>Graphic</td>
<td>Specification of the Shared Context in a collaborative task, the division in visualization areas and the finalization policy.</td>
</tr>
<tr>
<td>Interaction Modeling</td>
<td>Interactive tasks decomposition tree (CTT notation enriched with icons for specifying visualization areas in collaborative tasks)</td>
<td>Graphic</td>
<td>Interaction Modeling at individual responsibility level; interaction with the shared context in collaborative tasks.</td>
</tr>
</tbody>
</table>

Table 4
Roles involved in the Conference Review System.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCChair</td>
<td>The PCChair is responsible for creating the conference, determining the conference topics (or tracks) and subjects, establishing the Program Committee, and, advised by PCMembers, defining the final list of accepted and rejected papers. The conference is supposed to have a set of tracks (or topics) and, optionally, a set of subjects. The PCChair also defines the conference deadlines: submission, review, and notification.</td>
</tr>
<tr>
<td>PCMember</td>
<td>A PCMember is responsible for evaluating a set of papers assigned to him. A PCMember may indicate another person as a reviewer of a paper. The PCMembers are responsible for advising the PCChair to determine the final list of accepted and rejected papers.</td>
</tr>
<tr>
<td>Reviewer</td>
<td>Each reviewer is responsible for reviewing papers.</td>
</tr>
<tr>
<td>Author</td>
<td>An author is responsible for submitting paper/s for acceptance at the conference.</td>
</tr>
<tr>
<td>CoAuthor</td>
<td>For each paper, data about the coauthors is stored, i.e. those authors (if any) other than the author registered as the contact author.</td>
</tr>
</tbody>
</table>

has been made, each PCMember receives an email message with the list of papers, and a URL to a page where he can access these papers.

- A PCMember may re-assign a paper to a Reviewer. In this case, the reviewer will be responsible for entering the paper evaluation, but the PCMember is the final party responsible for the paper evaluation during the selection process. Each time a reviewer enters a review, the associated PCMember is notified of this event by the system. A PCMember, or a Reviewer, may enter a review for a paper assigned to him. Each paper review is made up of a set of evaluation items, each of which may be graded in the range 1-5. One of these items is a final recommendation, also in the range 1-5, where 1 means Strong Rejection; 2 means Rejection; 3 means Indifference; 4 means Acceptance; and 5 means Strong Acceptance.

- With the exception of the PCChair, all users must register in the system in order to participate in all processes. PCMembers are pre-registered by the PCChair, but must confirm their registration personally, including changing their password and entering affiliation and contact information. PCMembers may also indicate their preferences for conference topics and subjects, when defined. Reviewers may be pre-registered by the PCChair or by any other PCMember, but must also confirm their registration in the same way as PCMembers. Reviewers may not indicate any preferences regarding topics or subjects. Authors must register before submitting any papers. An Author may pre-register a CoAuthor, in which case the CoAuthor must register himself as well, in order to have access to the system.

- Once the review deadline has been reached, the review process is closed. All PCMembers are allowed to see all the papers and their respective reviews. The system is able to show the list of papers ranked by a number of relevant criteria, for instance when ordered by any of the evaluation items (including final recommendation), author, title, affiliation, subject, etc. After deliberation, the PCChair, together with PCMembers, chooses the papers that will be accepted and rejected,
marking them appropriately through an interface furnished by the system. During this selection process, the PCChair and PCMembers may visualize the status of the process at any stage, by looking at the list of papers already marked for acceptance, for rejection, and still undecided. Several statistics are also shown, such as totals for each category, acceptance rates, number of papers per track, per status, etc. Once the process is marked as finalized by the PCChair, the system issues a notification message to the authors of the papers, which includes the reviews submitted by the PCMembers and Reviewers. It is also possible to visualize the list of accepted and rejected papers.

The modeling process of the Conference Review System follows the stages shown in the previous section. Starting from Fig. 1 in which the stages of the proposal were presented, and adding the products produced in each of these stages, the overall process to be followed by the designers is shown in Fig. 3.

4.1. Sociogram

As indicated previously, the sociogram is a diagram that allows the organizational structure to be represented, as well as the relationships that can exist among its members. The elements that can make up this diagram are shown in Fig. 4.

In the example we identify the following roles: PCChair, PCMember, Reviewer, Author and CoAuthor. Fig. 5 shows the structure of the organization described in the problem statement (Table 4). A PCMember can be considered as a specialized Reviewer, since he carries out the same work (to revise) but specializes in carrying out another wider group of tasks or responsibilities. Once the inheritance (generalization/specialization) relationships among roles are established, the actor–role–acting relationship is added for each of the roles in the diagram. This kind of relationship can be labeled when we want to express cardinality (minimum and maximum) in the cases in which the specification establishes restrictions on the matter. The diagram can also include relationships among roles that can, at a given time, work together. These relationships are expressed by means of association relationships. In Fig. 5, we can see that PCChair and PCMember roles are associated. This indicates that there are tasks in which both participate with their respective responsibilities.

4.2. Responsibilities modeling and process modeling

The two following stages can be approached interchangeably in any order, since the information which contributed to the models generated in each stage is supplemented with the information that is specified in the other one. Even so, they have to be revised in a cyclical way. In the Process Modeling stage (that shows the interaction/collaboration that takes place
among the group members), creating the so-called participation table is of great help (see Fig. 7). Using this specification technique, of a textual nature, allows the designer to have an initial idea about the division of the work at the highest level of abstraction. This table is comprised of as many rows as tasks that have been identified at a higher level of abstraction by the designer and of as many columns as roles that were identified in the previous stage. A cell \((T_i, R_j)\) will be marked when the role \(R_j\) is implied in the accomplishment of the task \(T_i\), with the possibility of several roles taking part in the same task (group work tasks); as well as that, a certain role can take part in several tasks throughout the whole process. Once the appropriate cells are marked, the last column is filled. This allows the tasks identified to be classified into three categories (cooperative, collaborative and individual tasks), making use of a different icon for each type (Fig. 6).

From the problem statement of our case study we have identified six tasks in the highest level of abstraction that are involved in the collaborative process: Conference Creation, Defining Conference Features, Paper Submission, Review Distribution Task, Review Process and Making the final list of papers. These six tasks comprise the rows in the table. The five roles previously identified make up the columns of the table (PCChair, PCMember, Reviewer, Author and CoAuthor). For each task, we mark the roles involved in its execution. For example, the task Defining Conference Features is performed jointly by the PCChair and the PCMembers. This task is performed between them in a collaborative way. The same thing occurs with the Making the final list of papers task, in which these two roles are also involved. On the other hand, the Review Process task, in which the PCMember and Reviewer roles are involved, is considered a cooperative task due to the fact that there is a distribution of responsibilities between these two roles to achieve the aim of reviewing the papers. The PCMembers re-assign the papers to Reviewers, who are responsible for doing the paper evaluation and, finally, the PCMember is the final party responsible for the paper evaluation considering the Reviewers' opinions. Also, we identify two individual tasks: the Conference Creation task that is performed by the PCChair and the Paper Submission task carried out by the Authors.
In the process of construction of the participation table, we have detected some important characteristics of the system that we are modeling and it has helped us to understand it better. These characteristics are summarized hereafter:

- The role CoAuthor does not have an assigned responsibility. This indicates that it could be eliminated from the model representing the organizational structure.
- We have identified tasks of a group work nature (which imply the participation of several roles): Conference Creation, Review Distribution Task, Making the final list of Papers.
- Roles with a greater number of responsibilities, within the whole process, are detected (this is the case of the PCMember or PCChair).

Once the participation table has been constructed, we will center on the Responsibilities Model definition (see Figs. 8–11). Information expressed by means of the previous techniques serves as a basis for the definition of the responsibilities model associated with each of the system roles. Taking a reading by columns (by roles) in the previous table, we complete the tasks that each role must carry out, adding those that are of an individual nature and are not encompassed in the group work processes of the organization. So, for example, a new task of an individual nature entitled Paper Updating is added to the responsibilities model of the role Author. This task is an individual responsibility of the role but is not essential for specifying the group process. This kind of task has to appear in the responsibilities models, but not necessarily in the process model. By means of filling in the responsibilities models, we can create a list of responsibilities for each role, indicating, for each one, their type (individual task, task carried out collaboratively or cooperatively). For each task, the objects manipulated are specified including the access modifiers to these objects (R, Reading; W, Writing; C, Creation; and any combination of these). It is important to point out that the objects specified in these models are those that are necessary in the context of the interactive tasks (related to interaction between the final user and the user interface; at presentation layer in three-tier
architecture [33]) and not necessarily at the data model level (specified at persistent storage layer). The data manipulated at the human–computer interaction level can be a subset of the object specified at the data model level [8]. Also, for each task the prerequisites, which permit satisfactory execution, are defined. The prerequisites make reference to the tasks that should have been completed before the current one and also, that object/s in the data model must have been previously created by some role in the system. In this way, we can establish temporal execution dependences (order relationships) as well as data dependences among the main tasks.

Once the main tasks that characterize the group work and the responsibilities for each role have been defined, we will create the process model. This model allows the complete operation of the group process to be specified, which can be cooperative, collaborative or mixed. This model uses a graphical diagram that allows all the information defined by means of the two previous techniques to be related. This diagram is represented by means of a graph whose nodes are the identified tasks and the arcs are the transitions (dependences) between tasks.

Each task (see Fig. 12A) is represented by means of a node in the shape of a rounded rectangle that contains three parts with the following information:

- The head of the node includes the task name (on the left) and its type (on the right). To indicate the type we use the icons displayed in Fig. 6.
- On the lower left, the roles involved in the execution of this task are enumerated.
- On the right, the objects manipulated in the task are shown, preceded by access modifiers (R, W and/or C) at a task level. The specification of the access modifiers at role level will be made in later stages of the methodology, in which the abstraction level of modeling goes down.
The transitions of the graph allow the dependencies that can exist between the tasks to be detailed. These can be the following:

- **Temporal dependencies** (see Fig. 12B). Basically these allow order relationships among tasks to be detailed (precedence and concurrence). Arrows that connect the different nodes of the diagram could be labeled by one of the following temporal operators (subset of the temporal operators used in CTT notation): • > (sequence), ||| (concurrence) and > (deactivation).
- **Data dependencies** (see Fig. 12C). These indicate that data is passed between the tasks. CTT (notation whose operators we take as reference) allows this situation to be specified by means of the indicator [ ]. However, we are interested in showing the object/s that is/are transferred from some tasks to others in a graphical way. This circumstance is expressed by means of the name of the transferred object between the brackets that precedes the temporal operator.
- **Notification dependencies** (see Fig. 12D). A task originates a notification event that precedes the execution of a second task. This notification can be targeted to one or several roles. For indicating this circumstance, the icon that represents the notification can be accompanied by its names.

- The execution of a task, which can be simple or iterative, can depend on the fulfillment of a certain **precondition**. This precondition can be of a different nature, as indicated below:
  - **Completion dependences**. Situations can be represented where the task can be executed a certain number of times (n), or an indeterminate number that depends on the evaluation of a certain termination condition (?: condition). The expression of this condition is included in the definition of the task itself (node in the diagram).
  - **Period dependences** (see Fig. 12E). These indicate situations in which the beginning of a task depends on a certain quantity of time lapsing. Graphically, this circumstance is indicated in the flow arrow that exists for a concrete task, by means of the symbol &c, followed by the details of the temporal restriction.
  - **Execution dependences**. These allow situations to be expressed in which the execution of a certain task is optional. This optional nature depends on whether a certain condition related to the state of some object/s of the data domain is true at the time the task must begin. This type of precondition is associated with the task itself.
  - **Mixed dependences**. These occur as the result of the combination of the previous types.

Fig. 12 shows the process model associated with the system used in the example. It consists of six tasks at a high level of abstraction. As we have already identified, two of them are individually executed (those labeled with numbers 1 and 3), two of them are cooperative, with a clear division of subtasks that will be indicated in later stages of refinement (those labeled as 4 and 5). Finally, there are two tasks that are collaborative (labeled as 2 and 6). We can see how, in these two last types, we must specify the roles involved in their execution, whereas in the individual tasks only one role must appear. For all the tasks, the objects manipulated are indicated, preceded by the corresponding access modifiers. In this case, the work flow is sequential (the temporal operator used is >>).

The process begins with the Conference Creation task, performed individually by the **PCChair**. In this task, two objects are created in the system: the Conference object that includes all the information related to the event (name, dates, topics, etc.) and the PCMember object, which will have as many instances as the number of members in the program committee (2 minimum and 20 maximum, as we have indicated in the sociogram; Fig. 5). This task is followed by the Defining Conference Features task, in which specific information about the conference is specified. For this, the object created earlier (Conference) is accessed for reading and updating. In this case, the task is performed collaboratively by the PCChair and the PCMembers. Next, the PaperSubmission (task 3) by the Authors can be performed. We can see that this task execution must be concluded before the deadline for submitting papers is reached (temporal condition Conference.submissionDeadlineDate), the transfer of these papers being necessary ([[>]]) for the following task (Review Distribution Task). This task (number 4) is cooperative, with both the PCChair and the PCMember roles involved in its execution. It is necessary to indicate that the objects that are manipulated during the execution of a task (in this case, the objects PCMember and Paper) must be created in one of the previous stages of the process. Checking this circumstance allows the coherence of the model created to be validated. Once the execution of the task has concluded, the PCMember should be notified about the decisions that were made. For executing the following task (Review Process), it is necessary to use the papers as they stand after the execution of the previous tasks. Once the Review Process has finished (temporal condition Congres.reviewDate is reached) the reviewers’ decisions (included in the paper object) are used to make the final list of papers (task 6). Two roles are involved in the execution of this last task (PCChair and PCMember), that work collaboratively to create the FinalPaperList. Finally, when Conference.notificationDate is reached a notification is sent to the authors.

### 4.3. Group work task modeling

In this stage, the level of detail with which the previously identified group work tasks (collaborative or cooperative) are specified increases. It is important to highlight the necessity of modeling the cooperative tasks and the collaborative ones in different ways. The outstanding information in each one varies. Considering Dillenbourg’s definition (see Section 1), this distinction is translated into two important aspects: the **division of tasks** (in individual tasks in the case of cooperation) and in the manipulated **objects** (which are shared in the case of collaboration).

Furthermore, this distinction between tasks is directly involved in the configuration and appearance of the user interface created for supporting each of these two kinds of tasks. So, for example, in **cooperative work** between the team members, two different situations are possible: (a) those in which various team members handle the same objects, producing various
versions of the same object, which makes it necessary to come to a final decision-making process to decide which version will be the group solution, or (b), those in which various team members handle different parts of the same shared context. In this situation it is necessary to assemble individual contributions (partial solutions) to form the group solution. In this case, the responsibilities of each member (tasks to perform) are not only independent from the others’, but also different, since these affect different sections of the information shared by the group. Again, those tasks are followed by a decision-making process, which can be developed collaboratively or carried out by a specific role (in whose user interface this action must be supported). In each of these situations, the tasks to be supported by the user interface and the objects or sections of objects to be visualized are different. In the first case, the user interface is common for all members of the work team (the task to be carried out and the data to be visualized are the same). In the last case, the tasks can differ and the visualization of information is also different (the data to be shown is not the same). Then, the user interface for supporting each member’s work must also be different.

On the other hand, the user interface for supporting a collaborative task has to visualize the same shared objects, must better control access to the information, allowing the enabling and disabling of certain areas of the UI and, finally, show the update produced in the data visualized (the shared context). Also, in this kind of UI we have to include awareness mechanisms [10] that help to reduce the metacommunicative effort needed to develop collaborative activities, promoting real collaboration between the members of the group.

Next, as an example, we will show in detail the specification of a task of each type. In particular, we will show the modeling of the cooperative task **Review Distribution Task** and the collaborative task for the elaboration of the **Final List of Papers**.

### 4.3.1. Cooperative task modeling: Review distribution task

Cooperative task modeling uses the so-called **Responsibilities Decomposition Graph**. In Fig. 13, the model of the task that allows papers to be distributed for their later review is shown in detail. On the left we can see the roles involved and the objects manipulated. On the right the **responsibilities decomposition graph** appears. The notation used is similar to that used in the process modeling stage. In this way, we maintain a certain level of coherence in the notations. The elements that comprise this graph are tasks (represented as nodes) connected by means of directed arcs that show the work flow. Arcs can be labeled to express dependences (notification, period, etc.) as we have pointed out in previous sections. The nodes of the graph represent individual tasks in which a single role is involved (see Fig. 13).

At this level the objects manipulated in the cooperative tasks are specified, associating them with the roles involved. The objects manipulated by each role in a cooperative task can be:

- **Instances** of a subset of the data model (object or set of objects) or several parts of this data set (being able to increase the granularity to attribute level). This situation can appear in concurrent tasks that manipulate an object.

- **Versions of the same object or objects**. This situation implies the existence of a temporal relationship (non-concurrent) among the tasks that are part of the cooperative work process.

At this level, the access modifiers associated with the objects manipulated in the context of the cooperative task are also specified. This information allows us to obtain the **access control matrix** that relates the members of the work group and the objects shared in the task. The aspect of this matrix for the considered example is shown in Fig. 14. This table has as
Fig. 14. Access Control Matrix, can be extracted when the task is specified.

Fig. 15. Icons for representing visualization features and exclusive access to shared context.

many rows as the number of objects manipulated in the cooperative task, and as many columns as the number of roles that participate in its execution. We can see how the Paper object is accessed for reading and writing by the PCChair and PCMember roles. Since access takes place in different stages, as we can see in the responsibilities decomposition graph (Fig. 13), we can also see that it is a case of manipulation of different versions of the same object of the data model.

In this level, we can specify the previous matrix at the level of object attributes. It makes sense when the roles manipulate data items of smaller granularity than the object. Moreover, the detail level also increases in the definition of responsibilities assigned to the roles. Thus, a matrix similar to the one in Fig. 14 can be created, in which roles are related to the operations that can be carried out on the data model object.

4.3.2. Collaborative task modeling: Making the final list of papers

To explain this stage, we take as an example the task that allows creating the final list of papers. Modeling collaborative tasks implies knowing the roles involved in its execution and the data model objects that are manipulated in a shared way. For this, the specification of this kind of tasks is based on the definition of the shared context, that is, the set of objects that is visible to the set of users and the actions that can be performed on them [11]. Fig. 16 shows the appearance of the specification of the collaborative task Making the final list of papers. As in the cooperative task specification, the area on the left shows the roles involved, the objects manipulated and the access mode to these objects (reading, writing and/or creation). In this case, the roles involved in the task that allow the creation of the final list of papers are the PCChair and the PCMember. Both of them access the Paper object, but only the PCChair role is in charge of creating the FinalPaperList. The central area shows the data model objects manipulated that constitute the shared context. In the example, these objects are the FinalPaperList that groups the Papers selected and the Review associated with them. It is important to point out that the shared context could not have a direct correspondence with the data model objects in the application. More precisely, we can say that it is a view of the set of objects that have importance in the context of the collaborative activity. That is, the objects related to interaction and coordination in the context of the collaborative task. For specifying the shared context we use UML notation, to which we add some icons to express visualization features and block the objects that comprise the shared context (see Fig. 15). We consider that the information that comprises the shared context can be divided into one of the following areas: objects and/or attributes manipulated in the collaborative visualization area, those that are manipulated in the individual visualization area and those that make up the segment of exclusive edition (a subset of the data model accessed exclusively by only one application user at a time). In our case study, we decide that the FinalPaperList must be visualized by all the users involved in the collaborative task (users playing the PCChair and PCMember roles) and, therefore, the user interface must visualize it. However, each member can decide to visualize a specific Paper in their own UI. But only one of them can modify the Review associated with a certain Paper at a time. The Review object is considered an exclusive access area of the shared context. When a user is updating a Review, the UI area for the rest of the group members that allows it to be edited must be enabled.

This division into visualization areas can be different in each case. It could happen that a model does not include an area of individual visualization. This indicates that we are in a situation in which all the members that collaborate see exactly the same objects. If, in addition, they all see the information in the same exact way, we would be specifying a situation in which the visualization is governed by the technique of strict WISIWYS (What I See Is What You See) [35]. These strict situations can be relaxed, for example, using isolated work spaces for each user.

Finally, in the specification of the collaborative task, we include on the right side of the model (Fig. 16) the interaction tree in CTT notation associated with the collaborative task. This model is created in the interaction modeling stage (see Section 4.4). The card shown in Fig. 16 provides a complete view of the specification of an interactive collaborative task.

Like what occurred in modeling cooperative tasks, granularity can be increased at the attribute level and the responsibilities can descend to the operation level on the objects. The technique used to specify both situations is based on the use of a table and textual descriptions. In both cases, there are as many columns as roles involved in the collaborative
Another of the aspects that is defined when we specify a collaborative task is the way in which its finalization is agreed. It is not necessary for cooperative ones since, in these, a subtask guided to coordinate the members of the team should exist explicitly. The objective of this task will be to assemble the partial results and/or to agree upon the finalization of the task.

As for the finalization policy of the collaborative task we identify three situations:

- **Completion by individual responsibility.** When one of the roles has the responsibility to end the collaborative task.
- **Completion by shared responsibility.** When several roles, in combination, decide to terminate the task. In these cases, the policy of decision-making can vary (democratic agreement, etc.).
- **Supervised completion.** In this case the system determines when the task must finalize (for example, the system can decide to conclude it when the product generated has a certain level of quality, or when all their component elements are completed, or any other criterion related to the product generated or the process developed to obtain it).

Each of these circumstances is represented graphically by means of an asterisk (*) next to the role icons. In the first case, the responsible role is the one that has the (*). If there is a shared responsibility, all the roles involved are marked with (*). If none of the roles has the indicator of being responsible for the termination, this is a case of supervised completion. As we have shown in Fig. 16, this example is a scenario of completion by individual responsibility. As we can read in the problem statement, “the process is marked as finalized by the PCChair”.

### 4.4. Interaction modeling

In this stage, the designer is centered on specifying the dialog that can take place among the users (individual users) and the applications (user interfaces) that intervene in the collaborative process defined. Thus, for each individual task, an interaction model must be created both in the initial phase and during the refinement of the cooperative tasks. To create
these models, the designer should identify the tasks that are initiated by the user when he interacts with the application (interactive tasks) and those that are carried out by the computer (internal processes or information visualization; that is, application tasks). In addition, for modeling the application dialog, temporal order among the tasks must be specified.

To model the interaction, a notation exists that is broadly diffused in the community of Human–Computer Interaction. This language is CTT [25,26], which we have already mentioned. Using this notation, the models built present a hierarchical structure in tree form that permits several levels of abstraction to be represented. Using CTT, we can reach high levels of detail in the interaction model. This facilitates obtaining the final design of the user interfaces. CTT is oriented to cooperative tasks and shared context. We propose using this notation to model the interactive aspects. In addition, in the particular case of collaborative tasks and using the models that we have constructed in previous stages, the CTT tree that models the interaction can be generated directly. For this, the definition of the shared context and the information related to visualization and lock issues are used (Figs. 5 and 16). For this we apply the following process, which we present for the Making the final list of papers task of the example considered:

1. The root of the interaction tree (see Fig. 18A) uses the icon that represents collaborative tasks. In the example, we have labeled the root with the name of the collaborative task (Making the Final List of Papers).
2. The first level of the tree will be comprised of two tasks, related by the operator | (deactivation):
   1. The first one will have the icon used for indicating collaborative visualization and will group the rest of the interaction tree (see Fig. 18B).
   2. The other task is the one in charge of providing the completion of the collaborative process. In our case, this task is carried out by the PCChair.
3. In the third level, two tasks are added. The first task groups the subtree that allows the attributes of the shared context object that are visible for all roles to be displayed (in the example, the list of Papers). The other task facilitates access to the operations allowed in the area of collaborative visualization.
4. The fourth level contains the operations that any member of the group can carry out (operations of the shared object). By way of example, we have expanded the subtree of the operation called orderByEvaluation(). This structure will be repeated for each of the operations that the group can carry out (orderByAuthor(), calculateStatistics(), etc.). We indicate it by the generic task name Collaborative Operation i, in Fig. 18. The subtree has an interaction task on the left that allows the user to start the execution of an operation (in the example, to Choose “orderByEvaluation”). This operation will be executed by the system (task of application type orderByTitle()) which will later show the result of the execution (task To Show orderByTitle()). The task that facilitates access to the individual workspace is located at this same level. In our example, it will be the operation that allows access to an individual Paper on the list.
5. In the fifth level, we locate the root of the subtree that models the interaction which takes place in accessing the object that we defined for individual access in Fig. 16 (this subtree is shown in Fig. 18C). Each paper on the papers list comprises the area of individual visualization. All users who collaborate have access to the papers list, but each of them could visualize a specific paper individually. Several users may visualize the same paper privately at the same time. If we observe Fig. 18, we can see how the previous structure is repeated (from the levels 1 to 4). This generic and repetitive structure has been extracted and is shown in Fig. 19. This structure is practically a behavior pattern that can be identified in the work spaces in collaborative applications [20].
6. Among all the possible actions that the user can take in the individual area, one allows segment modification of the shared context exclusively. The subtree that allows visualizing this exclusive area has a specific icon in its root (see Fig. 18D). In the example, the review object can only be modified by one user at a time. The subtree that is constructed for this task has a subtree of application tasks on the left-hand side. These tasks display the information about the segment of exclusive access. The right part of this subtree has tasks that allow the user to introduce modifications to the object being manipulated. Later, storing these modifications will be confirmed or cancelled, as applicable.

The generated interaction tree is common for all roles. However, matching the information specified in the table of operation permissions on the shared context, this tree can be pruned to obtain an instance for each role.

The interaction model described is added on the right of the card shown in Fig. 16. Thus, this card has a complete view of a collaborative task (in this case Making the final list of papers) (Section 4.3.2). The task is completely specified using several abstraction levels. The lower level is specified using CTT notation. From this model the user interface can be obtained semiautomatically using proposals such as [18,24]. Fig. 21 shows a possible user interface for supporting the collaborative task studied. This is obtained using the techniques proposed by Paternò, in particular, the calculation of the Presentation Task Sets (PTS). Using PTS we can obtain the task set that must be enabled (and, in the context of the user interface design, visualized) at one time. Using the information expressed by the PTS, the type of interaction and application tasks, the features of the object manipulated and some guidelines for preserving usability, we can select the most appropriate user interface techniques (widgets) for supporting the interaction task model.

As we have said previously, we have enriched the CTT notation with three new icons that represent three visualization areas (see Fig. 15). These icons are used separately as roots of the subtrees in the interaction tree in CTT notation: (a) the subtree which represents the interaction with the shared context that is common for all group members involved in a multiuser task (collaborative visualization); (b) the individual interactions of each member in the group (individual interaction icons).

---

2 References to publications that have used CTT: [http://giove.cnue.cnr.it/CTTpublications.html](http://giove.cnue.cnr.it/CTTpublications.html).
Fig. 18. Interaction model associated with the collaborative task *Making the final list of papers*. It is generated from the definition of the shared context.

Fig. 19. Repetitive CTT structure for modeling the interaction in the different spaces (of collaborative visualization, individual visualization and of access for exclusive modification).

*visualization*); and (c) the subtree which specifies the dialog with the area of the shared context that can only be accessed exclusively by one member of the group at a time. Using our extension of CTT, we can identify additional information about the areas that comprise the collaborative user interface. Thus, this extension has higher-level semantics which better organize and express specific interactive issues for collaborative applications.

The PTS extracted from the CTT model corresponding to each of the three visualization areas are shown in Fig. 20:

In Fig. 21, we can see the appearance of the user interface for supporting the collaborative task *Making the final list of papers*. Also, an enriched version with awareness techniques (Fig. 21B) and the web version (Fig. 21C) are shown. In Fig. 21A and 21C the PTS associated with all the UI widgets are specified.
Fig. 20. Presentation Task Sets in the CTT model.

It is important to indicate that, from the underlying data model of the collaborative task, we are obtaining an amodal interface. However, if we only consider the division of tasks into subtasks, and the temporal order among them, we would obtain modal interfaces. Using modal interfaces we obtain a good representation of the navigation, but we do not consider the manipulated data. One of the desirable features in an user interface, from a usability viewpoint, is that it must be easily controllable by the user. It is easier to reach this objective if we have amodal dialogs like the ones we obtain by applying our proposal.

5. Discussion

In the previous section, we presented the use of CIAN notation for expressing collaborative and interactive aspects of groupware applications. This is done at different abstraction levels. In this way, we can center on high-level models for a better understanding of collaborative issues or, alternatively, we can concentrate on low-level models. The latter are derived from the former in order to express details about the user interface that needs to be designed to support the human–computer interaction in the development of collaborative tasks.

CIAN notation is used in a methodological approach called CIAM. This methodology indicates how to use the notation for obtaining the Graphical User Interface of a collaborative application from the specification of the supported tasks. This feature is not directly supported in other methodologies of software development which are more widely known, such as the Rational Unified Process (RUP).

Comparing our proposals with other approaches in the same terms in which they were analyzed in Section 2, we can conclude that our proposal supports most of the features considered in Tables 1 and 2:

- CIAM can be considered a mixed approach that focuses on different modeling perspectives in several stages of the process. The first stages of CIAM tackle a more group-centered modeling, changing to a more process-centered modeling in subsequent stages, and finally draw closer to a more user-centered perspective as we go deeper into the abstraction level.
- The creation of CIAN notation is based on a metamodel (ontology) [21]. By means of the definition of our conceptual framework, we clarified the concepts to be managed and modeled, as well as the relationships that exist among them. This ontology served as a basis for the definition of the modeling techniques that are used in each of the stages of our methodological approach. It captures the concepts whose semantics are interesting to represent by means of the group of views or modeling perspectives to be tackled during the different stages. This conceptual framework includes concepts related to interactive and collaborative issues.
- We consider the most outstanding contribution of this proposal to be the fact that it distinguishes between cooperative and collaborative tasks and can generate the interaction model from a shared context definition.
- Unlike the rest of the studied approaches, CIAM considers interactive and group work issues jointly.
In relation to the **modeling of group work aspects**, CIAN notation allows specification of group work processes that are comprised of cooperative, collaborative and individual tasks performed by roles that manipulate objects and shared data to jointly reach the objective of the organization.

As a result of the proposed process, we obtain a set of CTT models that can be used as input in the existing approaches for deriving a concrete and final user interface.
Fig. 22. Modeling of cooperative and collaborative tasks using CIAT.

The CIAM methodology is supported by a CASE tool that allows editing and validating the models specified using CIAN notation. This tool is called CIAT (Collaborative Interactive Application Tool). Fig. 22 shows a screenshot of CIAT. This tool supports the interface design of groupware applications, enabling integration with software processes through the mapping of models in CIAN notation onto UML models [22,13]. Eclipse Framework provides tools for guiding the modeling tool development by means of using metamodels [23]. Eclipse supports provide for MDD (Model Driven Development) by means of EMF (Eclipse Modeling Framework). The CIAT tool has been implemented using EMF and GMF (Graphical Editing Framework). Using these technologies and the conceptual framework from which we created our methodological approach [21], we have created the CIAT tool as an Eclipse plug-in. CIAT provides designer models in XMI files. The XML Metadata Interchange (XMI) is an OMG standard that facilitates model interchanges via XML Documents.

6. Conclusions

The number of applications used to support the work carried out by different people who can play different roles is increasing. Additionally, the user interface of these applications is progressively receiving greater attention, since it can be a determinant when users accept or reject an application. Consulting the research dealing with both problems, we detected the lack of a proposal that allows combining interaction, collaboration and information sharing aspects. There are no proposals of notations and specific conceptual models that allow modeling for the peculiarities in this kind of systems. This paper presents a methodological approach to solve this shortcoming. It is comprised of a series of stages in which conceptual models are created using CIAN notation. These stages start by modeling the organization which will use the application to design, as well as the tasks to be supported by the collaborative system. The proposal starts from a high level of abstraction specifications and, in the following stages, the level of detail increases. In the last stages, the interaction between the individual users and the application is modeled. The different stages are connected and guide the designer until the CTT model is created. This notation, used in the interaction modeling stage, acts as a bridge in order to obtain the final user interface.

We have shown the application of the CIAN notation and our methodological approach by means of an example. We have managed to model and consider specific aspects of collaborative and cooperative tasks. These models have led directly to the design of the supporting user interface and interaction.

Unlike the rest of the studied proposals, CIAM makes a clear distinction between interaction within the group, and the individual interaction that occurs between a user and the tool he uses. Furthermore, unlike other approaches which only tackle individual and/or cooperative modeling, our proposal handles the modeling of purely collaborative aspects. Therefore, we can say that CIAM provides more complete support with a greater capacity to represent semantics than the rest of the proposals that deal with the design of the presentation layer in CSCW systems.

We are presently working on integrating the CIAM proposal and its notations with the standard UML in the context of the CIAT tool. Also, a great number of experiences in using the CIAN notation are being analyzed with the aim of validating the usability of the notation, the agility of the proposed method, and establishing the utility and position of CIAN notation in comparison with other existing approaches (in particular, UML and CTT). Our future work will consider the improvement of our proposal by including aspects related to mobile computing and context modeling.

Acknowledgements

This work has been supported by the University of Castilla – La Mancha, the Ministerio de Ciencia y Tecnologia and the Junta de Comunidades de Castilla – La Mancha as part of the mGUIDE (PBC08-0006-512), E-CLUB II (TIN2005-08945-C06-04) and M-CUIDE (TC20080552) projects.
References