The Journal of Systems and Software 86 (2013) 1772–1789

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Metamodel-driven definition of a visual modeling language for specifying interactive groupware applications: An empirical study

Ana I. Molina[a,*], Jesús Gallardo[b], Miguel A. Redondo[a], Manuel Ortega[a], William J. Giraldo[c]

[a] Department of Information Technologies and Systems, University of Castilla-La Mancha, Spain
[b] Department of Computer Science and Systems, University of Zaragoza, Spain
[c] System and Computer Engineering, University of Quindío, Quindío, Colombia

ARTICLE INFO

Article history:
Received 1 November 2011
Received in revised form 30 April 2012
Accepted 14 July 2012
Available online 31 July 2012

Keywords:
Groupware design
Interaction Design
DSL
Metamodel
MDE

ABSTRACT

This work is framed in the area of software development for *Computer Supported Cooperative Work* (CSCW). These software systems are called *groupware* systems. The development of groupware systems is a complex task, a problem that can be addressed applying the *Model Driven Engineering* (MDE) principles and techniques, where the use of models is essential. However, there are no proposals to address all issues to model in this kind of application (group work, shared context, coordination, etc.) and, in particular, there are no proposals that consider the modeling of both interactive and collaborative issues. To solve this deficiency, a *domain-specific language* (DSL) called *Collaborative Interactive Application Notation* (CIAN) has been proposed. To define this DSL a *metamodel* has been created describing the universe of discourse of the applications supporting interactive group work. We have defined the syntax and semantics of this language. We have also implemented a tool (called CIAT) for supporting the edition and validation of models created with CIAN. This tool has been implemented using the metamodeling facilities provided by the *Eclipse* platform. Finally, an empirical study was conducted with the aim of verifying the suitability of this approach and the perception of software engineers about its usefulness. The results obtained show that our proposal can facilitate the development process of groupware systems.

1. Introduction

The use of applications framed within the paradigm of *Computer Supported Cooperative Work* (CSCW) is increasing. Besides, the correct development of the user interface (UI) of this kind of applications deserves special attention, since it determines, to a large extent, its acceptance by the user (Grudin, 1994; Shneiderman and Plaisant, 2010). Therefore, the design of these aspects (cooperative behaviors, human–computer interaction issues and the existence of shared information workspaces) must be properly supported (Ellis et al., 1991; Johnson, 2004; Preece et al., 2011).

Regarding the development of CSCW systems we find three main approaches: (a) the ad hoc development, in which the systems are built completely tailored to the specific problem; (b) the use of *toolkits* and APIs, which provide higher level implementation facilities, as Habanero (Chabert et al., 1998), Dream Team (Roth and Unger, 1998) or Groupkit,[1] among others, remark; and (c) the development of CSCW systems based on *components* (Hummes and Merialdo, 2000; Slagter and ter Doest, 2001; Marsic, 2001) and *patterns*[2] (Lukosch and Schümmer, 2006; David et al., 2003; Guerrero and Fuller, 1999; Isla et al., 2006), which allows the design and construction of CSCW systems by using predefined building blocks and designs that can be reused and combined. On the other hand, recently another approach is gaining attention: the design and development based on conceptual modeling of collaborative applications. This line of work can be considered as framed within the *Model-Driven Engineering* (MDE) paradigm (France and Rumpe, 2007). This paradigm has become widely used, achieving great importance in *Software Engineering* (SE). In general, we can say that a software development process incorporates the principles of MDE when this makes it possible to conceptually separate the modeling space and the implementation space. Applying the principles of MDE improves the quality of the software systems, the degree of reuse and, as a result, implicitly, the efficiency of their development (Bezivin et al., 2005). Therefore, we consider that the field of *Groupware Design* can consequently benefit from the application of

* Corresponding author. Tel.: +34 926 295 300; fax: +34 926 295 354.
E-mail addresses: AnaIsabel.Molina@uclm.es (A.I. Molina), jesus.gallardo@unizar.es (J. Gallardo), Miguel.Redondo@uclm.es (M.A. Redondo), Manuel.Ortega@uclm.es (M. Ortega), wjgiraldo@uniquindio.edu.co (W.J. Giraldo).

[1] http://www.groupkit.org/.

[2] http://www.groupware-patterns.org.

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.07.049

MDE techniques and methods. We intend to demonstrate it in this paper.

Studying the existing alternatives for CSCW systems modeling (Molina et al., 2009a) we noticed certain deficiencies in modeling collaborative aspects. These problems arise specially in proposals that combine both group work application aspects and interactive aspects. These problems confirm the lack of an appropriate notation for designing (and modeling) interactive and collaborative applications. To try to solve the problems pointed out we took under consideration the definition of a notation that would allow specifying the most important aspects of an interactive groupware system as fully as possible, taking into account different perspectives. In the definition of that visual language we try to follow the principles of MDE. As for the developing of modeling languages, MDE proposes two alternatives: the use of *General Purpose Languages* (for example, UML), which can be used to solve a wide variety of problems, or the use of *Domain Specific Languages* (DSLs). The latter has been the approach chosen because of the expressiveness of DSL in contrast to *General Purpose Languages*.

When designing a DSL we must identify what the DSL should express (Völter, 2009). In this sense, the use of a *domain definition metamodel*, which is a specification of the domain conceptualization, can be considered compulsory to obtain the notation (Kurtev et al., 2006; Henderson-Sellers, 2007; Moody, 2009). To create this metamodel, here we will revise the main conceptual frameworks in the area of CSCW and *Human–Computer Interaction* (HCI). Once the conceptualization identified, we will define a *notation* (CIAN; *Collaborative Interactive Interaction Notation*). Finally, and by applying the principles of MDE, we will translate this notation into a formal language and implement suitable editors to support the modeling of interactive groupware application (Völter, 2009).

The remainder of the paper is organized as follows. In Section 2 an overview of related work is provided. Section 3 presents the main aspects of the CIAN notation, including the abstract syntax (based on a metamodel), the concrete syntax (a collection of icons that constitutes the graphical language) and the constraints on the abstract syntax and the semantics. In Section 4 the tool implemented to support the edition of CIAN models is briefly described. Following, in Section 5 we explain two empirical studies carried out for assessing the acceptance of this interactive groupware requirements modeling method based on user perception. Finally, in Section 6 conclusions and future work are outlined.

2. Background and related work

This section provides a brief review of the most representative works related to the application of the principles of MDE in the areas of groupware and in the field of HCI. Below we will present a review of the main conceptual frameworks in both fields. There will be a comparative analysis of the concepts considered by each of these proposals. This analysis will facilitate the creation of the *metamodel* as a basis for defining our notation. Finally, we present related work in the field of assessing the quality of modeling languages (DSLs).

2.1. Model-driven engineering principles in HCI and CSCW areas

The term *Model-Driven Engineering* (MDE) is commonly used to describe software development approaches in which the creation of abstract models of software systems are systematically transformed in concrete implementations (France and Rumpe, 2007). MDE proposes the use of *models* describing the system from different points of view or perspectives and considering different levels of abstraction. Currently, MDE focuses primarily on developing techniques, methods, processes and support tools that reduce the gap between the problem and the implementation (Frankel, 2004). The application of the principles of MDE implies that modeling languages have certain levels of specification by means of a syntax and semantics formally defined (Frankel, 2004). A widespread language to represent language syntax is MOF (*Meta Object Facility*, from OMG).[3] MOF can be used as a base language for the specification of other notations. Moreover, this formalization facilitates the manipulation of these models through tools, as well as checking the semantic validity of these models (Favre, 2005). Environments and processes framed within the MDE must provide adequate support for the manipulation of the models and for their storage. Therefore, it is necessary to implement a series of editors that facilitate the creation of models, as well as support to the transformations that can occur between them.

The area of HCI has applied the principles of MDE mainly on the development of user interfaces (UIs) (Vanderdonckt, 2008), leading to the area called *Model-Based User Interface Development* (MBUID). MBUID (Sousa et al., 2007) proposes the use of models to identify, organize and reflect about the components and the behavior of an interactive system. MBUID offers different levels of abstraction, reaching the use of *task models* its highest level (Vanderdonckt and Berquin, 1999). Task models are logical descriptions of the tasks required by the user to achieve their goals while interacting with the application (Dix, 1993; Paternò, 1999). The use of task models in the process of designing and developing UIs allows building more user-centered interactive applications. In addition, there is a good communication tool between the various people involved in the development of a system (designers, engineers, users, experts in the domain, etc.) (Diaper and Stanton, 2004). Paternò (2001b) provides a review of the main notations for declarative modeling of the interactive aspects of applications. *ConcurTaskTrees* notation (CTT) (Paternò, 2004) is certainly the most widespread and accepted notation for *task modeling*. This notation is practically considered as a *de facto* standard in the HCI community.

Also, the modeling of group work has been of interest in the areas of HCI, as well as in Software Engineering (SE). Among the most relevant contributions in the field of HCI we can highlight the *Group Task Analysis* (GTA) framework (van Welie and van der Veer, 2003), the CUA (*Collaborative Usability Analysis*) notation (Pinelle, 2004) and a task analysis method called MABTA (*Multiple Aspect Based Task Analysis*) (Lim, 2004). In the fields of CSCW and workflow systems, we can find the *Action Port Model* (APM) notation (Carlsen, 1998), which is taken as reference by notations such as RML, TaskMODL and DiaMODL proposed by Trætteberg (2002), the DCWPL (*Describing Collaborative Work Programming Language*), a coordination programming language that allows developers specifying collaborative applications, decoupling coordination from computational issues (Cortes and Mishra, 1996; Cortes, 2000), and the *Proclets* proposal (van der Aalst et al., 2001). As for approaches derived or completely framed within SE, we can find two extensions of UML notation: COMO-UML (Garrido et al., 2005) and UML-G (Rubart and Dawabi, 2004).

A more detailed review and comparative evaluation of different notations for interactive and collaborative modeling can be found in (Molina et al., 2009a).

2.2. A review of conceptual frameworks of CSCW and interactive systems

As we pointed out, the use of declarative models in the development of CSCW systems is considered a useful tool. However, there is no agreement among authors on the set of concepts and abstractions that should be included in such models. In order to propose a

[3] http://www.omg.org/mof/.

conceptual framework or *metamodel* that serves as the basis for the definition of a notation that allows the full specification of these applications, we will make a brief tour of some of the main proposals in this field. The objective of this analysis is to identify the concepts common to all of them, their strengths and weaknesses, in order to propose a suitable conceptual framework. This framework will be enhanced by a series of new concepts and relationships that address the modeling of the interactive aspects in such applications.

One of the frameworks that served as the basis for most of the proposals in the field of CSCW is the *Activity Theory* (Kuutti, 1991). From the point of view of computation this theory explicitly addresses the interaction between users using distributed architectures for joint activities (which would lead to computational support to the CSCW paradigm). The Activity Theory takes into consideration the sociocultural aspects of knowledge and action and focuses on the concept of *activity* as a basic unit of analysis. On the other hand, the *Coordination Theory* (Malone and Crowston, 1990; Crowston et al., 2004) is a set of principles for the management of *interdependencies* among activities to be developed to achieve an objective. This theory focuses on coordination problems that can occur, such as goal identification, the association between goals and activities, the ordering of the activities, the selection of actors to perform an activity, the management of interdependencies between activities and the allocation of resources to an activity. Another relevant work is GTA (*Group Task Analysis*) (van der Veer and van Welie, 2000), whose conceptual framework proposes a set of representations for the specification of different aspects: the *work structure*, the *workflow*, the *work artifacts* and the *environment* in which group work takes place. One of the highlights of this proposal is the definition of a conceptual framework (van Welie et al., 1998) that underlies the definition of these models as well as further validation. This conceptual framework defines five basic concepts (*task*, *object*, *role*, *agent* and *event*) and a set of relationships between them. The *team-enabled workflow reference model* (van der Aalst and Kumar, 2001) includes and relates concepts from *groupware* and *workflow* technologies. These authors propose the use of OCL (OMG, 2003) to express constraints with respect to the distribution of work to teams using OCL. The conceptual framework proposed in the methodological framework AMENITIES (Garrido and Gea, 2001; Garrido, 2003) stands out from other proposals, giving special importance to such aspects as the organization, the group and their behavior. Finally, we highlight the proposal TOUCHE (Penichet et al., 2010), a novel approach to gather requirements for groupware applications. TOUCHE provides a set of requirements templates that include new information to give account of specific features concerning groupware applications. The creation of these templates is based on a metamodel (Penichet et al., 2007).

After reviewing the most important frameworks in the field of CSCW systems design the main concepts and their relationships have been compiled (Table 1).

Most of these frameworks focus their models on three main aspects: the *work* to be done to achieve a certain goal, the *agent* or *subject* that performs such work and the *resources* used to carry it out. Table 2 shows some of the concepts related to these aspects that can be considered as equivalent in the different proposals. We added a new column in which we group other items not related to any of the three aspects above. Noting the concepts considered in this last column we can see that they are related to the *constraints* imposed by the environment in which teamwork develops the work and with the *coordination tasks* of such work (allocating resources to tasks, allocating roles to tasks, ordering of activities, etc.).

Other aspects that are observed when comparing all these proposals are:

- As for *relationships*, with the exception of activity theory, all the frameworks support the decomposition of tasks or activities into smaller units of work. GTA also allows the representation of the decomposition of the managed objects into simpler objects and the definition of subroles. This is expressed by the reflexive relations highlighted in bold in Table 1.
- As for the modeling of the *issues of coordination* between work activities, GTA, the team-enabled workflow reference model and AMENITIES consider the relationship between tasks, and the *triggering* between them. However, only the Coordination Theory goes into more detail by providing different types of interrelationships between tasks.
- Activity theory, AMENITIES and TOUCHE take into account the characteristics of the *community* or *organization* within which group work is developed. They make use of the concepts of *rules* and/or *restrictions* to include these features in their specifications.
- Although some of the proposals include the modeling of *Tools* supporting group work, none of them addresses the modeling of the interactive aspects of these tools. Only GTA notation evolved later in order to support interactive features.

There are few proposals presenting conceptual frameworks for the specification of the *interactive aspects* of the applications. One of the approaches most relevant to the issues addressed in this paper is the one proposed by Limbourg (2004) that presents a set of metamodels associated with the main models used in a MBUID process: the conceptual view of the *task model*, the *domain model*, the *abstract user interface model*, the *concrete user interface model* and the *context model*. It also proposes a framework that links all these models and serves as a basis for defining the rules of transformation or mapping between them (Limbourg and Vanderdonckt, 2009). van der Veer and van Welie (2000) proposed an extension of the basic GTA ontology that relates task modeling with dialog modeling expressed in UAN notation (Harstson and Gray, 1992). Finally, in (Mori et al., 2004) the authors presented the metamodels associated with the CTT notation.

As a result of this comparative analysis we can conclude that although COMO-UML and TOUCHE seem to be the most complete of all the proposals studied, these approaches leave some issues incomplete: shared context specification, notifications and interactive and collaborative visualization issues. Considering the features identified during the study of these approaches, as well as what some of them lack, we consider it necessary to propose a notation that guide software engineers in the modeling process of interactive issues in groupware applications.

2.2.1. Reflection: the notation requirements

The study of the different modeling proposals in the areas of HCI and CSCW, and the comparative analysis of the aforementioned conceptual frameworks allowed us to enumerate the requirements of the visual language that we intend to define in relation to the limitations noted. They are:

- Such language should include the *modeling of the organization* and the identification of *roles.*
- It should support the *different models* of *interaction between team members* (i.e., the specification of group work) and *individual-level interaction* of each member (interactive aspects).
- Models will be proposed to specify the main issues to consider in the group work, i.e. the work to do, the agents or subjects that perform this work, as well as the resources they use. The notation will show graphically the relationship between these three aspects.
- As to group work, we will consider the *task* as the basic unit of analysis, so that the specifications to create are *task models.* We distinguish *task* in the context of group work from the so-called

Table 1
Comparative table of the main conceptual frameworks for developing CSCW systems.

Framework	Concepts handled by the framework	Relationships	Highlights
Activity theory	Activity (A), Action (Ac), Community (Com), Subject (S), Rule (Rul), Tool (To), Object (O), Division of Labor (Div), Result (Res)	S-O (mediated by To), S-Com (mediated by Rul), Com-O (mediated by Div), O-Res	• Inclusion of rules and tools
Coordination theory	Objective (Obj), Activity (A), Actor (Act), Resource (Re), Interdependence (I)	Obj-A, Act-A, **A-A**, Re-A	• It includes three types of *interdependencies* between activities (*prerequisite, shared resource, simultaneity*)
GTA conceptual framework	Task (T), Object (O), Role (R), Agent (Ag), Event (E)	E-T, **T-T**, T-O, **O-O**, R-O, **R-R**, Ag-R, Ag-T, T-R	• Serves as a basis for a method of task analysis • Inclusion of the *event* concept • Detailed definition of features
Team-enabled workflow reference model	Role (R), Task (T), Team (Te), Resource (Re), Activity (A), Work_Item (WI), Case (Ca), Contribution (C), Team_Position (TP), Team_Type (TT)	R-Re, Te-Re, C-Re, **R-R**, **T-T**, T-A, WI-T, WI-Ca, TP-R, **TT-TT**, TP-C, A-Ca	• Combination of *groupware* and *workflow* technologies concepts • Inclusion of *team* concept • Use of *OCL* to express *constraints* with respect to the distribution of work to teams
AMENITIES conceptual framework	Strategy (S), Objective (Obj), Organization (Org), Role (R), Context (C), Group (G), Actor (Act), Restriction (Rest), Law (L), Capacity (Cap) Task (T), Activity (A), Action (Act), Event (E), Object information (O), Artifact (Art)	S-Obj, S-Org, Obj-Org, Org-R, R-T, R-R, A-A, T-T, Rest-R, G-R, A-T, Act-O, O-Art, Art-G, Art-Org, C-Org, C-G, C-Art, C-O E-Act, E-Art, E-O, E-G, E-Org, etc.	• *Organization* modeling (inclusion of concepts of *law, capacity*, *strategy*, *objectives* and *context*)
TOUCHE conceptual framework	Role (R), Task (T), Objective (Obj), Actor (Act), Restriction (Rest), Law (L), Capacity (Cap), Group (G), User (U), Agent (Ag), Session (S), Protocol (P), GroupTask (GT), Coordination (Coord), Communication (Com), InformationSharing (IS), Time (Tim), Place (P), Cooperation (Coop), Collaboration (Col)	R-T, T-Obj, A-R, R-P, R-S, G-L, Act-R, S-A, A-Cap, A-Obj, A-G	• It includes concepts related to main taxonomies in CSCW (3C: *coordination, communication, collaboration* and Johansen's Time-Space Matrix (Johansen, 1988)) • Inclusion of concepts *session*, *information sharing* or *protocol*

interaction task (in the context of work of an individual user with the application).

– The notation should allow specifying in a distinctive way the *cooperative* tasks of the purely *collaborative* ones. None of the approaches studied supports the distinction between collaboration and cooperation. In many contexts, these terms are often used as synonyms. Dillenbourg et al. (1996) clarifies the difference 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 (Ellis et al., 1991). *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 result obtained 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 (the *shared context*) (Ellis et al., 1991). Taking 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.
– The language should support the modeling of different levels of abstraction and, in particular, the *decomposition of tasks into subtasks*.
– It should be able to specify aspects of *coordination* (one of the basic concepts in CSCW systems).

Table 2
Equivalence of concepts across the conceptual frameworks discussed.

Framework	Work to be done	Agent or subject who performs the work	Resources used	Other concepts
Activity theory	Activity (A), Action (Ac)	Community (Com), Subject (S)	Tool (T), Object (O), Result (Res)	Rules (Rul), Division of Labor (Div)
Coordination theory	Activity (A), Objective (Obj)	Actor (Act)	Resource (Re)	Interdependencies (I)
GTA conceptual framework	Task (T)	Role (R), Agent (Ag)	Object (O)	Event (E)
Team-enabled workflow reference model	Task (T), Activity (A), Work_Item (WI)	Role (R), Team (Te), Team_Position (TP), Team_Type (TT)	Resource (Re), Contribution (C)	Case (Ca)
AMENITIES conceptual framework	Strategy (S), Objective (Obj), Task (T), Activity (A), Action (Act)	Organization (Org), Role (R), Actor (Act), Group (G)	Information object (O), Artifact (Art)	Event (E), Law (L), Capacity (Cap), Restriction (Rest), Context (C)
TOUCHE conceptual framework	Task (T), Objective (Obj), GroupTask (GT), Cooperation (Coop), Collaboration (Col)	Role (R), Actor (Act), Group (G), User (U), Agent (Ag)	InformationSharing (IS)	Restriction (Rest), Law (L), Capacity (Cap), Session (S), Protocol (P), Coordination (Coord), Communication (Com), Time (Tim), Place (P)

- The models that specify the work to be done will allow specifying the *workflow*, i.e. the *order* in which tasks should be developed.
- It will consider the *qualitative temporal information* modeling and the *quantitative* one (Lacaze and Palanque, 2004). 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, and choice. Quantitative temporal information is related to delays, dates and durations.
- It will also need to support the modeling of *information passing* between tasks as well as triggering *events* and the sending of *notifications*, enhancing in greater detail the *workflow* specification.

Considering these requirements we have defined a notation called CIAN (*Collaborative Interactive Interaction Notation*) (Section 3). However, if practitioners are to consider adopting this modeling method, they should know how suitable and effective it is. In the next subsection we present the main contributions that address the issue of evaluating DSLs.

2.3. Methods for assessing DSLs quality

One important aspect to consider when a new DSL is proposed is to evaluate the quality of its concrete syntax (Moody, 2005; Nelson et al., 2005). In this field there are two main approaches. One that proposes the use of *objective* metrics to measure certain characteristics of models (Fenton and Pfleeger, 1997) and those that propose the use of *subjective* ratings to measure issues as *perceived ease of use* of models, *perceived usefulness* or *intention to use* the proposed languages and models (Mayer, 1989).

In the first approach we find some relevant contributions. Some of them address the definition of the *quality attributes* to measure (one of the main issues to concrete when assessing the quality of a notation). Thus, Kolovos et al. (2006) propose as the general requirements of a DSL the *usability*, *conformability*, *orthogonality*, *supportability* and *simplicity*. Most of the rest of contributions in this field take as the basis the ISO 9126 quality model (ISO/IEC, 2001). This standard proposes two main characteristics: the *usability* and the *maintainability* of languages. *Usability*, in this context, is defined as a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users. Usability is measured from other three sub-characteristics: *learnability*, *understandability* and *operability*. The other quality characteristic is *maintainability*, defined as a set of attributes that bear on the effort needed to make specified modifications. Maintainability includes three sub-characteristics: *stability*, *analyzability*, *modifiability* and *testability*. In the literature we also find proposals of frameworks (Nelson et al., 2012) and metrics (mainly based on structural properties of diagrams) to assess conceptual data models in an objective manner. Most of these proposals have been applied for assessing UML diagrams (Genero et al., 2005; Saeki, 2003).

Some authors propose guidelines for defining DSLs (Karsai et al., 2009). Thus, in Moody (2009) the author defines a set of principles for designing cognitively effective visual notations. In Kelly and Pohjonen (2009) the worst practices for domain specific modeling are presented. All these guidelines and principles can be used not only to construct visual languages but also to evaluate, compare and improve them.

Finally, we find contributions that propose methods for evaluating the quality of modeling methods based on *comprehension* of models (Gemino and Wand, 2005; Mayer, 1989) and *user perceptions* (Maes and Poels, 2007; Abrahao et al., 2011). These authors propose assess visual languages in a subjective way. All these evaluation methods intend to verify the suitability of DSLs (and predict the acceptance in practice), based on the effort of applying a particular modeling method, the perceived quality of the artifacts produced, and the user perceptions with regard to the quality of the method. In this paper, we apply this last approach for the empirical evaluation of the quality of the proposed DSL (CIAN) (Section 5).

3. The Collaborative Interactive Application Notation (CIAN)

The next subsections summarize how our visual language (CIAN) has been developed by following the tasks defined by Feilkas (2006). First, we present the initial conceptualization built to define the universe of discourse of interactive and collaborative applications (Guizzardi et al., 2002; Guizzardi, 2005). We use UML notation to represent the metamodel, because this is appropriate to specify its two main aspects: *concepts* and *relationships* (Guizzardi et al., 2004; Li, 2005). The framework is divided into different *views* that represent different perspectives from which we can analyze the system (Dijkman et al., 2003). For each of these views we can make use of any of the representations in the literature or, if necessary, we can propose new ways to represent each view. Once the conceptualization (the metamodel) identified, we define the constraints on the abstract syntax, which describes in precise terms the way to structure the constructors of the proposed DSL.

Definition of the abstract syntax (*domain definition metamodel*): A conceptual framework for modeling of interactive issues in groupware

The CIAN notation has been developed from a metamodel created for this purpose. This metamodel defines the *abstract syntax* of CIAN, and provides a conceptual framework that encompasses the elements previously identified as relevant to model groupware applications. Fig. 1 shows the UML representation of the proposed metamodel. This conceptual framework has the following features:

- For each of the aspects to model mentioned in the previous section (*work* to do, *subject* that performs it and *resources* available to do it) we define a *view* of the conceptual model: the *process view*, the *organizational view* and the *data view*, respectively.
- As we mentioned, the fourth concept considered in most proposals is the *tool* concept. Regarding the tools, we are interested in the concepts related to *interaction*. Therefore, we add a fourth view that includes the concepts related to this issue: the *interaction view*. We understand *interaction* as the actions that occur between users and the system for the use of an interactive application and for any purpose (Redondo, 2005).
- The concepts that form each of the views are interrelated, as well as there are relationships between concepts belonging to the four views.

Table 3 shows the **concepts** for describing all perspectives (or *views*) to consider when specifying interactive groupware systems.

The **organizational view** includes concepts related to the structure of the *organization* that will develop the group work. An *organization* is composed of the set of roles which the *groupware* tool to design is to support. This view includes the concepts of *role* and *actor*.[4] Usually an *actor* is considered a human agent. But a *group* of people or a *software agent* can also be considered actors. A *group* consists of several actors. An actor can play different roles and a role can be played by several actors. This view also includes the creation of specializations/generalizations of roles.

[4] *Actor* vs. *role*. We define an *actor* as an active entity (human or not, individual or group) that can play one or more roles at some point within the system. We call *role* to a set of activities or responsibilities that can be performed by one or more users of a system at a given time.

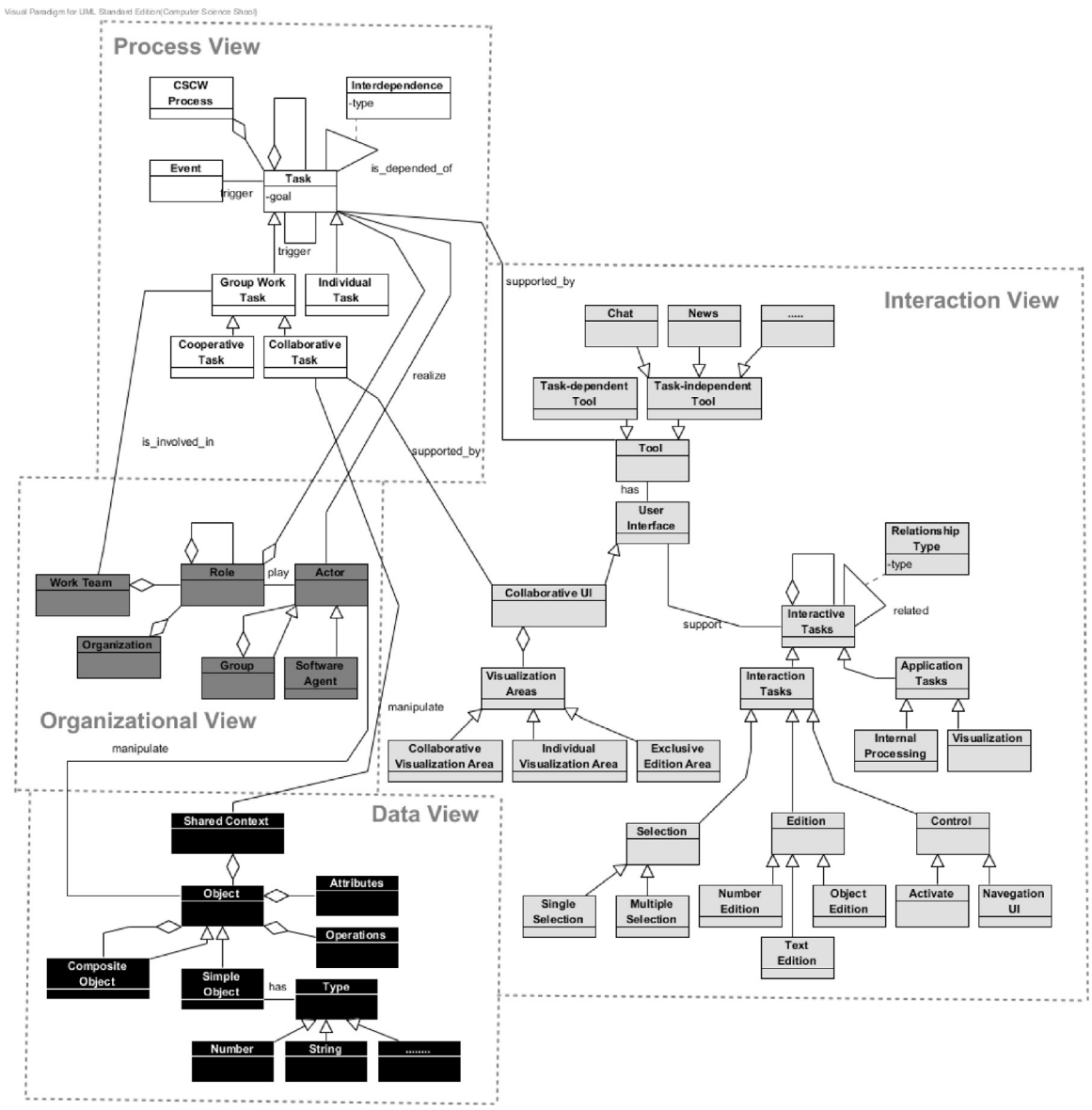

Fig. 1. Conceptual framework for the design of interactive *groupware* applications.

A *work team* is composed by a set of roles that work together in performing some group work task. Sometimes the concepts *group* and *work team* are used as if they were synonyms. However, there are some differences between them that should be clarified. A *team* is a specific type of group that has certain characteristics that distinguish them from ordinary groups. A team can be defined as: "*A small number of people with complementary skills who are committed to a common purpose, a series of performance goals and approach, all of which are mutually accountable*" (Katzenbach and Smith, 1999). In teamwork participation, collaboration and cooperation are promoted, so that the work is based on the expertise of all its members and the objects and results are shared.

The **process view** allows modeling the structure and the work flow, i.e. the *process modeling*. The main concept of this view is the *task* concept. A *task* is an activity performed by roles to achieve a certain *goal*. The *work structure* represents how people perform their work, usually by decomposing tasks into tasks of lesser difficulty until getting to more basic ones. The tasks, to achieve its objective, must be performed in a certain order. That is, in order to understand how a particular activity is carried out it is also interesting to know the *work flow*. These order relationships allow us to specify coordination and temporal relationships. To express order and temporal relationships between tasks we use the set of temporal operators included in the CTT notation, based on the set of temporal operators of LOTOS (*Language of Temporal Ordering Specification*) (Bolognesi and Brinskma, 1987), because they are probably the set of operators that can represent a greater number of situations. Another aspect of interest to be represented when modeling *work flow* is related to the *objects* that are passed between tasks, as well as the *events* or *notifications* that keep on the workflow dynamics. In general, these characteristics are included in the *interdependence* concept. An *interdependence* can be of different types: *temporal* (expressed by the aforementioned temporal operators), *data* (the sending of information between tasks) and *notification* (the sending of an event or a message). The *goal* concept is represented as an attribute of task because a goal only makes sense in the context of a task and does not depend on other items. Similarly, the *type* of interdependencies that may occur between tasks are represented by an attribute of the *Interdependence* association class.

One feature that differentiates our approach from other authors' is the distinction between *cooperative* and *collaborative* tasks. This

Table 3
Definition of the main concepts included in the conceptual framework.

Concept	Definition
Organizational view	
Organization	Social structure in a work environment consisting of a set of roles.
Role	A role is defined as a set of activities or responsibilities that can be performed by one or more users of a system at some point.
Group	Set of entities (actors who play roles) belonging to a same organization or participating in carrying out the tasks.
Software agent	Software entities that perform a set of operations on behalf of a user or another program with a certain independence degree or autonomy. In addition to being autonomous it should be reactive, social, and have its own initiative.
Actor	An actor is an active entity (human or not, individual or in-group), which can perform one or more roles in a given time within the system.
Work team	Set of roles working together in a group work task.
Process view	
CSCW process	Set of activities needed for achieving an objective within an organization. The process includes a specific organization of the group work activities. The process includes information concerning the communication and coordination necessary to carry out the work. It shows the flow of control from one activity to another.
Task	Activity carried out by the roles of an organization to achieve a certain *goal*.
Event	Occurrence of any fact that is taking place at a particular moment and that produces a change in the system state.
Interdependence	Relationships that can occur between tasks and define the workflow in the organization. These may be of several types: temporal, data and notification relationships.
Group work task	Task performed by a work team.
Individual task	Task performed by a single role.
Collaborative task	Group work task in which several roles take part working in a coordinated way on a common task, which produces a common result.
Cooperative task	Group work task in which a common aim exists, but in which the team members work in an independent way or in different areas of the common information space.
Data view	
Shared context[a]	Shared context is defined as the set of *objects* that are visible to the *work team*, as well as the *actions* that can be executed on them.
Object	Entity that contains the information necessary to complete the tasks, or that is generated as a result of them.
Attributes	Pairs (Identifier, Value) that define properties of the objects in the data model. They define the structure of the object.
Operations	Operations to be performed on the objects of the data *view*. They define the behavior of the object.
Composite object	Object of information composed of simpler objects.
Simple object	Minimal information unit in the *data model*.
Interaction view	
Tool	Software used by the users to perform activities and achieve their goals.
Task-dependent tool	Tools that support tasks directly related to the application domain.
Task-independent tool	Tools to support communication and coordination.
User interface	Contact surface between the user and the application.
Interactive task	It is defined as the logic activity necessary to achieve a certain objective, and to which the user interface should provide support. Depending on who performs this task in the human–computer dialog that takes place through the user interface, is classified into interaction tasks (initiated by the user) and application tasks (performed by the application).
Interaction task	Task carried out by the user during the interaction with the system. This kind of task is initiated by the user.
Application task	Task executed by the application in the context of the dialog that occurs between the user and the interactive applications.
Relation between interactive tasks	Precedence and information passing relationships between interactive tasks.
Collaborative user interface	Also called *Multiuser Interfaces*. This concept refers to user interfaces of *groupware* applications.
Visualization area	Sections in which the visualization (or displaying) of the *shared context* can be divided.

[a] *Environment* vs. *shared context*. We understand environment as the hardware and software that establish the context of the interaction. We should distinguish it from the so-called *shared context* (Ellis et al., 1991). The latter makes reference to a subset of the environment, formed by the set of objects (documents, etc.) and actions on them visible to a set of users.

distinction, which has been described in the previous section, influences the division of tasks as well as the objects manipulated and, therefore, they have different representations in the modeling of our system.

Fig. 1 shows the metamodel associated with the *process view*. In this diagram the main concepts that define both the decomposition structure of tasks and the flow between them are collected. A *CSCW process* is defined as a set of *tasks*. The execution of each of these tasks allows achieving a certain *goal*. The tasks are related between them by *trigger* relationships (flow between tasks) and *decomposition* of complex tasks into simpler ones. Tasks can be linked by *interdependencies* of different types. A task is performed by an *actor* playing a certain *role*, and to perform it certain information is manipulated (*object*). A task may be of individual or group work nature. In the latter case a *work team* (set of roles) is involved in its execution.

The **data view** includes the data manipulated during the work in group. This view is a subset of the *domain model* of the software application and includes *objects* that form the *shared context* and the objects manipulated by each of the tasks performed. An *object* includes the definition of a set of *attributes* and *operations* that can be performed on the object. There are two types of relationships between objects: *composition* (objects contained in other objects called *composite objects*) and *type* (objects can define a hierarchy of types).

The **interaction view** includes the concepts related to the *tools* and the *interactive aspects* of the application. There are several conceptual frameworks that include the concept of *tool* (which we define as the software used to support the performing of tasks). We can classify the tools into two types:

- *Task-dependent tools*. This concept includes all the tools in the *groupware* application that support the tasks directly related to the application domain.
- *Task-independent tools*. In the context of collaborative applications, communication, coordination and decision making are critical activities (Miao and Haake, 1998). We understand *communication* as the process of message exchange and *coordination* as a set of mechanisms used to establish a coherent link between the tasks of each user. *Decision making* will be used to seek consensus, to vote proposals and to support negotiations. It is necessary, therefore, to have software tools that support this

Table 4
Classification of *task-independent tools*.

Support tool	Communication support	Coordination support	Synchronous	Asynchronous
Electronic mail	X			X
News	X			X
Chat	X		X	
Agenda		X		X
Decision support system		X	X	

type of interaction task among the users. Some of these tools are generic and applicable to any *groupware* system regardless of its domain. Table 4 shows a classification of some of the most common support tools. The designs of groupware systems should consider the specification of these auxiliary tasks. In some cases these tools implement well known interaction patterns or protocols (Molina et al., 2006).

Each of the tools, especially those that are task (or domain)-dependent, has user interfaces that have to be modeled. The modeling of the user interface could be done at three levels (Vanderdonckt, 2005) (Fig. 2): (1) *Task and Domain Modeling*, (2) *Abstract User Interface Specification*, which represents the structure and content of user interface in abstract terms, that is, the information to be displayed in each window and the dialog needed to interact with that information, and (3) *Concrete User Interface Specification*, which specifies the representation style information units (Szekely, 1996).

We are interested in the highest level of abstraction and, in particular, in *task modeling*. The key concept in task modeling is the *interactive task* concept. *Interactive task* is defined as the logic activity necessary to achieve a certain goal, and which the user interface should support. The order and coordination of interactive tasks can be expressed by using *temporal operators*, which indicate the *type of relationship* that exists between them. When creating the interactive tasks model we can specify different levels of abstraction, i.e., the tasks can be decomposed into less complex tasks. This situation is expressed by the aggregation relationship shown in the metamodel associated with this view. The basic tasks that make up the model may be of two types (Paternò, 1999):

- *Application tasks*. They are executed by the application (i.e., information displaying or internal processing).
- *Interaction tasks*. They are carried out by the user when interacting with the system. We consider three *basic interaction techniques*: selection, editing and control. S*election tasks* allow the manipulation of *composite objects* (concept included in the data view). A selection task can be of two types: *simple selection* or *multiple choice selection*. *Edition tasks* allow the modification of objects of different types (*text*, *numbers*, etc.). *Control tasks* allow the *execution* of application functionalities and the *navigation* through the elements of the user interface.

The *User Interface* supports *interactive tasks*. A *Collaborative User Interface*, also called in the literature *Multiuser Interface* (Antunes et al., 1991; Antunes and Guimaraes, 1994; Koch, 1996), should be considered a specific type of *User Interface*. We consider that in a collaborative user interface we can identify three main visualization areas of the shared context. This way of structuring the user interface is associated with the use of the metaphor of the *workspace* used to organize the tasks to do (Pankoke-Babatz, 1994). Each workspace can be considered as a resource area with a specific functionality that supports the execution of a task or subtask. These spaces can be shared, when the task is performed by a group of collaborators, or individual, when they are used by a single user (Sarin and Greif, 1985). Therefore, we consider the following *visualization areas*:

- *Collaborative visualization area*. The user interface area that supports the display of the segment of the shared context that is accessible by all members of the group.
- *Individual visualization area*. The user interface area that supports the display of the individual workspace.
- *Exclusive edition area*. The user interface area that allows displaying objects that are accessed exclusively by one group member.

Maybe a model does not have an individual visualization area. This implies that there are some situations in which group members can visualize exactly the same objects. If, in addition, the information is presented exactly in the same way, we are in a situation in which the collaborative user interface is governed by the strict WYSIWIS technique (*What You See Is What I See*) (Stefik et al., 1987). Adding an individual visualization area involves relaxing the WYSIWIS, possibly by separating workspaces (the public space and a private area).

3.1.1. Relationships between views

Each view represents a different perspective to take when specifying a groupware system. However, these views are interrelated. The relationships between the concepts of different views are described below:

- As already seen in the other frameworks previously studied, it is necessary to relate the *work* with the *subject* who performs it and the *resources* manipulated to carry it out. So we have added the following relationships between the views:
  - Task–Actor–Object.
  - Task–Tool.
  - Role–Task.
- Regarding the features related to group work modeling, we can add these new relationships between the views:

- Group Work Task–Work Team.
- Collaborative Task–Shared Context.
- Collaborative Task–Collaborative User Interface. The group work tasks are supported by tools with a specific type of User Interface: the Collaborative User Interface.

In the same way that, at a single-user level, the concepts *Task–Actor–Object* are related, we relate, in the context of group work, *Group Work Task–Work Team-Shared Context.*

3.2. Definition of the concrete syntax: defining the visual language

Once the starting metamodel created, it is necessary to define a visual language for specifying the concrete syntax of each of the views identified. We begin by studying the possibility of using the diagrams provided by UML to model each of them. Following we

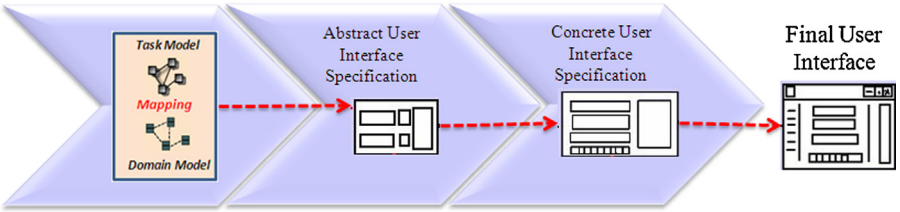

Fig. 2. Abstraction levels in user interface modeling.

present the proposed notation and justify the choice made to represent each of the views.

3.2.1. Considering the UML notation for modeling the metamodel views

UML (Rumbaugh et al., 2000) is currently the most widely used standard notation. Although UML does not consider the modeling of interactive tasks, it has been considered as an alternative for group process modeling (Eriksson and Penker, 2000). In Table 5 we can see which of their models may be more appropriate to represent each of the views of the proposed metamodel. We can see that the most appropriate notations to represent views in which dynamic behavior is modeled (process and interaction views) are the activity, state machine, collaboration and sequence diagrams. We can use the last two to represent interactions between roles (for representing collaboration/cooperation process) or between the final users and the system (to model interaction), but not between objects, which is its regular use. More static representations, concerning the organization structure and data manipulated can be represented in a more adequate way with class diagrams. Following we discuss about the potential use of each of UML diagrams for our purposes.

- The *use case diagrams* are an excellent tool for capturing high level requirements of the system and are used to delimit it and express the higher-level interactions that occur with the exterior of the system. They can therefore be very appropriate to understand the high-level functionality both of a group work process as well as of the more external interaction that occurs with the presentation layer of the application. This makes it a tool to express the highest level of abstraction of both the *process* and the *interaction* models. The *interaction* modeling focuses on specifying the tasks that the user should request the application (for reaching its goals) and the response that it generates. This description is somehow related to the requirements that the application should give support, expressed as a *use case diagram*. In fact it can be considered a step previous to the creation of the *task model* (Paternò, 2001a). In addition to this, these diagrams can express relations of specialization/generalization of actors, which shows part of the information interesting to capture through our *organizational view*.
- *Class diagrams* describe the structure of a system. The structure is specified by a set of classes and relationships. Classes can be used to represent and structure information, products, documents or even organizations. Therefore, the use of this representation is adequate to express the structures of two views in our framework: the *data view* and the *organizational view*. The use of these diagrams shows that both views are of static nature. Thus two of the concepts of our framework (*roles* and *objects*) can be mapped directly as classes in this type of diagrams.
- The so-called UML *interaction diagrams* can take two forms: *sequence diagrams* and *collaboration diagrams*. The former show one or more sequences of messages sent between objects, while the latter describe a complete collaboration between a set of objects. While the first term takes into account the sequentiality of the interactions, the second one gives more importance to delimit the role each object plays in its interaction with other objects. Both diagrams allow representing dynamic behaviors and are often considered semantically equivalent. These diagrams can be used to express the *process view* in which there are sequences of activities performed by different roles. Instead of expressing interactions between objects, these objects should represent system roles, showing the messages exchanged. Sequence diagrams are suitable to represent the sequence of tasks between roles. However, one of its major limitations is that they do not take conditional or optional situations into account. As for the *interaction* modeling it could be specified by a sequence diagram, which only shows the interaction between the *user* role and the *user interface* class. Operations and message passing can be interpreted as activation of tasks and information flows. Sequence diagrams are also suitable for *scenarios* modeling. The use of scenarios is a common technique in the HCI field and consists of creating informal descriptions of specific uses in the context of an application (Preece et al., 2011). Sequence diagrams support the modeling of a limited sequence of activities that define a usage scenario. That is, the objective of both techniques is similar.
- *Activity diagrams* describe activities and actions that take place in a system. This diagram is the most suitable to represent both *process* and *interaction models*. An activity diagram is a variation of a state machine where states represent the execution of actions and activities. The roles may also be represented by these diagrams dividing them into as many *swimlanes* as roles involved in group work. These diagrams allow specifying the control flow and the objects passed between the activities of the process described. They also allow representing conditional flows and concurrent executions, which, enhanced with an additional operator (iteration, etc.) could support the work flow specification. This alternative has been considered by other authors (Eriksson and Penker, 2000; de Cesare and Lycett, 2002; de Cesare and Serrano, 2006).
- *State machine diagrams* express possible states and transitions between states through which an object pass over its lifetime. This diagram can be used to represent manipulations of objects in the context of a work flow or the transformations that an object (for example, part of the shared context within a collaborative task) undergoes. For interaction modeling, state machine diagrams could be used to represent the dialog that takes place between the user interface and the user (Cox and Walker, 1993)

Table 5
Relationship between UML diagrams and views of the proposed conceptual metamodel.

UML diagram	Organizational view	Process view	Data view	Interaction view
Use case diagrams	X	X		X
Class diagrams	X		X	
Sequence diagrams		X		X
Collaboration diagrams		X		X
Activity diagrams		X		X
State machine diagrams		X	X	X

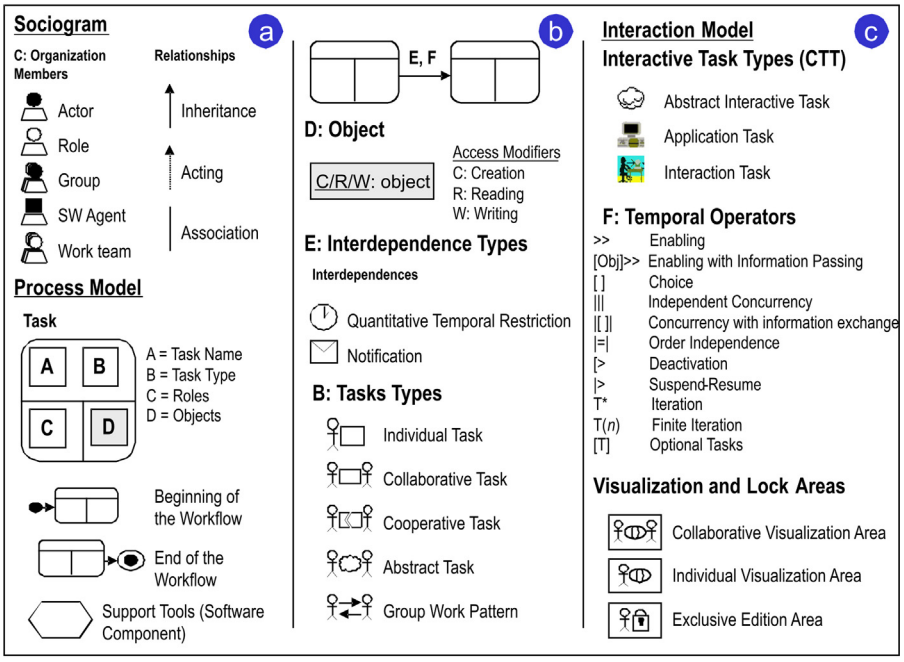

Fig. 3. CIAN notation summary.

or to model the navigation and activation of windows in the user interface (Markopoulos and Marijnissen, 2000). Other authors (Garrido, 2003; Garrido et al., 2005) propose the use of state machine diagrams, in the context of group work, to represent dynamic aspects of the organization.

As we can see, there are two UML diagrams suitable for *group work modeling*: use cases and activity diagrams. On the other hand, the *design of interactive systems* can make use of use cases and sequence diagrams. In terms of class diagrams, they seem appropriate for both.

We propose a new notation (called CIAN) that includes some models inspired in UML (class, state machine and activity diagrams). However, the models created with CIAN have different objectives from UML ones. Besides, the graphical representation proposed is more suitable for supporting the multidisciplinary *stakeholders* involved in groupware design, some of which have no specific knowledge about computer science (etnographs, sociologists, etc.).

3.2.2. Defining the graphical notation: CIAN

After studying the possibility of using UML as an alternative notation we proposed a more suitable representation to create our specification language. Thus, a concrete syntax was defined in order to make the language more usable. The metamodel was used as the basis for the definition of graphical icons that form the CIAN notation (*Collaborative Interactive Application Notation*). These icons were designed bearing in mind their usability when applied by users (in our case the multidisciplinary *stakeholders* involved in the design of a *groupware* system). Fig. 3 includes a representative sample of the elements of the CIAN models. On the top left (Fig. 3a) of the figure, we can see the icons that represent the organization members (roles, actors, software agents, etc.). In the bottom left (Fig. 3a) and the top center (Fig. 3b) areas, we can see the icons representing the nodes that form the *process model* and the ones indicating the different tasks and interdependence types. On the right-hand side (Fig. 3c), 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 the complete concrete syntax (CIAN) can be found in (Molina et al., 2009b).

As for the representation of the *organizational view*, we have chosen a representation showing the relationships of specialization/generalization between members of the organization, with arrows of relationship between roles added to specify other types of relationships (*acting* and *association*). The representation is similar to UML class diagrams, but we have chosen to use graphical icons representing the different types of components of the organization (*actors*, *roles*, *groups*, etc.). Still, the translation to a UML class diagram is relatively straightforward. Fig. 4 shows the structure of the organization in a *Conference Review System* (Molina et al., 2009b).

When addressing the modeling of the *process* we considered the use of UML activity diagrams with *swimlanes* for each role in the system. This first option was discarded because it had problems of expressiveness when trying to represent tasks that involved more than one role. This representation is suitable for representing *cooperative* tasks, in which there is a division of tasks between the different roles, but it cannot model the joint work

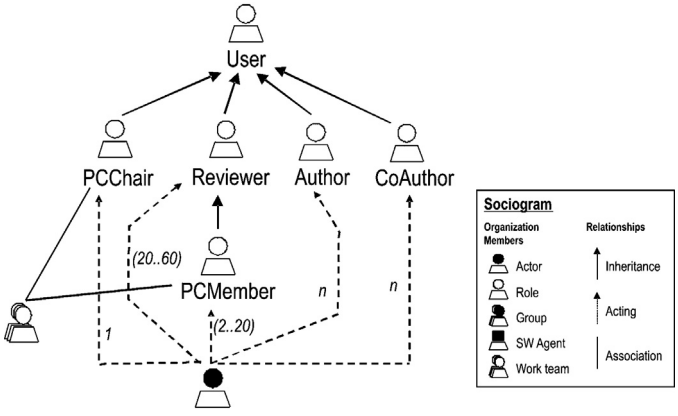

Fig. 4. Example of *sociogram*.

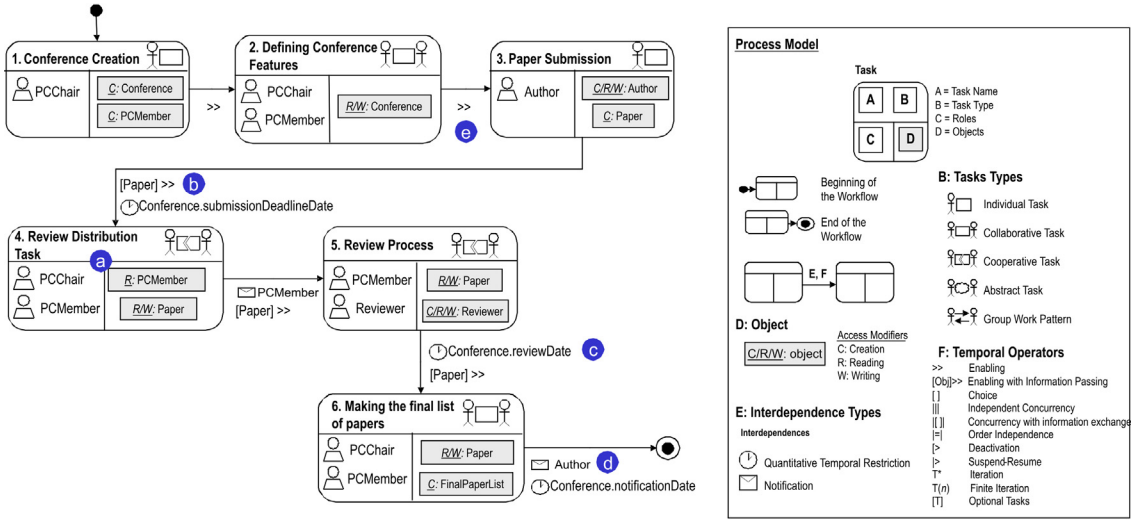

Fig. 5. Example *process* model.

or *collaboration* between roles adequately. Since activity diagrams are a special type of state machine diagram, we chose this second option. In this way we can represent, in each of the states that form the work flow (Fig. 5), the name of the task, the roles involved and the objects manipulated (Fig. 5a). In the arrows we represent the data transferred (Fig. 5b), the restrictions or conditions of the execution (Fig. 5c), the notifications (Fig. 5d) and the temporal operators (Fig. 5e). On the other hand, the notation should allow us to represent not only the *work flow*, but also the *work structure*, that is, the decomposition of tasks into subtasks. In the state-based modeling that we propose, the structuring of tasks into subtasks of lower complexity is addressed through the refinement of states into new diagrams of lower level of abstraction.

Each task must be classified in one of the following categories: *cooperative task, collaborative task* or *individual task*. *Collaborative Task* modeling requires the specification of the roles involved in its execution (Fig. 6a), as well as the objects of the data model manipulated and shared by the work team (Fig. 6b).

The modeling of the *data view* (which includes the *shared context*) is based on a *class diagram* in the standard UML notation. This view serves as a bridge to the rest of the system design, since the *domain model* (from which the data view is a subset) is created by designing the processing layer of the application (Schulte, 1995). 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) (Fig. 6b).

As for the *interaction view* (Fig. 6c), we chose the CTT notation, among the available notations, due to its graphical nature (making it more readable and usable), since it allows handling different levels of abstraction and includes broader temporal operators. In addition, this notation is taken as starting point for several MBUID systems for automatic or semiautomatic user interface generation (Paternò and Santoro, 2002; Luyten, 2004; Mori et al., 2004). Thus, for example, the TERESA application (Paternò and Santoro, 2002) allows us to derive user interfaces adapted to different devices from CTT task models. We propose to enhance this notation by adding three new icons that represent the three *visualization areas* considered by our conceptual framework. Each of these areas has a subtree of the overall CTT model associated (Fig. 6c): (a) the subtree that represents the interaction with the *shared context* that is common to all group members involved in the collaborative task (*collaborative visualization*), (b) the interaction of individual nature of each group member (*individual visualization*) and (c) the subtree that specifies the dialog with the area of the shared context that can only be accessed exclusively by one of group members at once. Using this extension of CTT notation, we can specify additional information about the areas that compose the *Collaborative User Interface*.

It is necessary to note that the three differentiated parts in which the shared context has been segmented are directly related with the subtrees identified in the CTT model. Modeling of this relationship between the objects manipulated and the interactive tasks is very useful for selecting of the best visualization technique in creating the User Interface (Pribeanu, 2002; Pribeanu and Vanderdonckt, 2002; Stocq and Vanderdonckt, 2004).

3.3. Definition of semantics: abstract syntax constraints

The use of metamodels, in addition to serving as the basis for the definition of a notation, has a second facet that can be of great utility. Their definition allows the formalization of a set of *axioms* that represent the constraints that affect the objects and predicates specified by the metamodel. This set of axioms provides a *declarative semantics* of the modeling technique that is created from that conceptualization. This formalization allows verification of the models created (Henderson-Sellers, 2011).

Following we show some of the predicates derived from the definition of our conceptual framework. We use the identifier *a* to denote the concept *actor*, *r* to denote *role*, *o* to represent an *object*, *t* to indicate *task* and *g* to represent the *goal*. In our ontology we define the following predicates: *perform (a, r); reach (g, t); manipulate (t, o); realize (a, t); responsible(r,t), etc.*

We also specify the specialization/generalization relationships between roles. We establish a new relationship *specialized_role(r, r′)* to indicate that *r* is a specialized role of *r′*. We also define the inverse relationship *generalized_role(r′, r)*, which is defined as: *generalized_role (r, r′) ≡ specialized_role (r′, r).*

A specialized role inherits the responsibilities of its parent role. This situation can be expressed by the following axiom: *specialized_role (r, r′) ∧ responsible (r′, t) → responsible (r, t)*

From the definition of the process view we can test the conditions under which the organization has reached a certain goal. We call *t* to a *task* t and $t_1, \ldots, t_n$ to the subtasks in which the task *t* is decomposed. We suppose the subtasks are related by selection

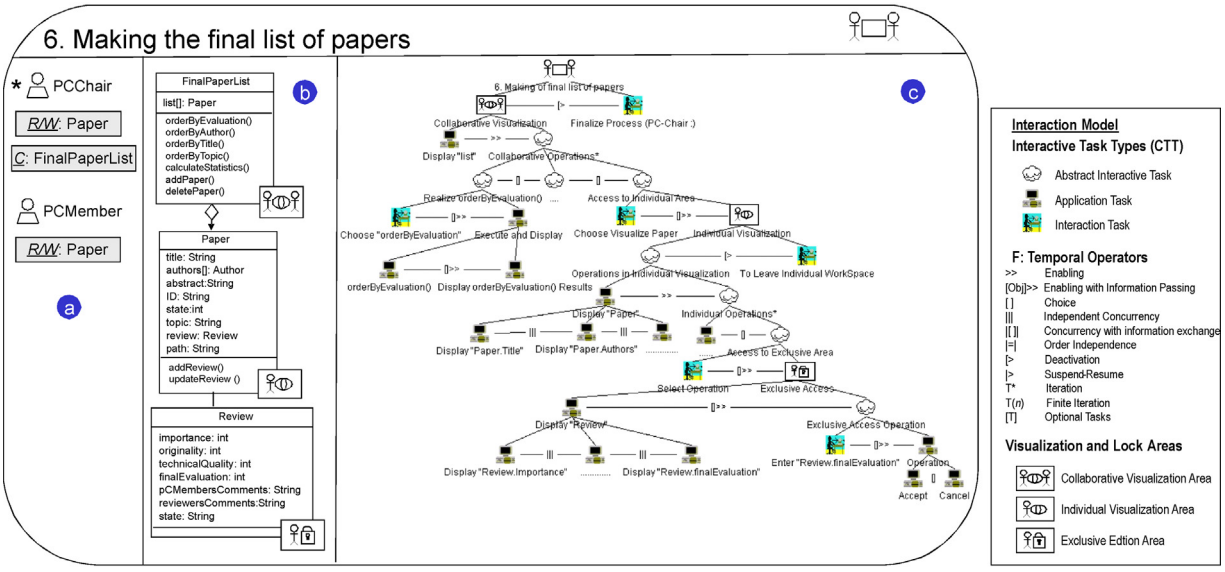

Fig. 6. Model of a *collaborative task*.

operators. In this situation, checking if a certain goal has been achieved or not is represented as follows:

reached_goal (g, t) ≡ *reached_goal (g, t_1)* ∨ . . . ∨ *reached_goal (g, t_n)*

The definition of a set of rules or axioms associated with our conceptual framework allows the validation of features about the structure and behavior of our system. We could ask questions as:

- *"What are the goals of the role R?" (?g contains the answer)*
  responsible (R, t) ∧ *goal (?g, t)?*
- *"What objects should be available for the role R to achieve the goal G" (?o contains the answer)*
  responsible (R,t) ∧ *goal (G, t)* ∧ *manipulate (t,?o)?*

The answers to these questions would determine if our model fulfills the constraints imposed by the underlying conceptual framework.

Therefore, the proposed metamodel expresses the semantics of the CIAN notation concepts, whose syntax is defined by the metamodel. The static semantics in the CIAN is composed of the OCL constraints which have been on the abstract syntax or metamodel. Finally, the editor of the CIAN notation (presented in the next section) provides the operational semantics of the language.

4. Supporting the edition of CIAN diagrams: the CIAT tool

In this section we introduce CIAT (*Collaborative Interactive Application Tool*), a model-based software tool aimed to support designers and engineers for the edition of the models in the CIAN notation. This CASE tool allows not only the edition of models in CIAN notation but also the validation of the correctness of these models (the tool implements the OCL constraints defined from the metamodel).

CIAT has been developed by using the *plug-ins* in the *Eclipse Modeling Project* (Moore et al., 2004; Pelechano et al., 2006). Specifically, the CIAT tool has been implemented as an Eclipse *plug-in* (Clayberg and Rubel, 2006) using EMF (*Eclipse Modeling Framework*),[5] GEF (*Graphical Editing Framework*)[6] and GMF (*Graphical Editing Framework*).[7] By using these technologies we have created the editor starting from the defined metamodel (which was previously specified in EMF). In Fig. 7 we can see an extract of the EMF representation (*Ecore* model) for the *sociogram* specification.

As mentioned above, we use OCL (OMG, 2003) to define the semantics of the language (Fig. 8). Some of these rules implemented in CIAT are:

1. Circular inheritance is not allowed: "*not self.supertype->includes(self)*".
2. A relationship cannot be related to itself: "*self.next != self.previous*".
3. A role cannot implement several actors: "*self.itsRelations-> select(r|r.type=Acting)->forAll(t1,t2|t1<>t2 implies t1.next<>t2.next)*".

In Fig. 9 we can see the appearance of the CIAT tool. On the left we see the creation of a *sociogram* using the editor (Fig. 9a). On the right we see the XMI file that provides persistence to the model created (Fig. 9b). The *XML Metadata Interchange* (XMI)[8] is an OMG standard that facilitates the interchange of models via XML Documents. CIAT uses XMI files to store the system domain models. In Fig. 9c and d we show the modeling of cooperative and collaborative tasks using the CIAT tool.

5. Two empirical studies with CIAN

In order to empirically validate the proposed notation, two experiences were carried out in the Computer Science Faculty at the University of Castilla-La Mancha, in Spain. In this section the details of these empirical studies are described and discussed.

5.1. Overview

This section describes two controlled experiments carried out to provide evidence with regard to the potential usefulness and usability of CIAN for interactive groupware modeling. We formulate the goal of our empirical study as follows (Wohlin et al., 2000):

To analyse models represented with CIAN
with the purpose of evaluating the notation,

[5] http://www.eclipse.org/modeling/emf/.
[6] http://www.eclipse.org/gef/.
[7] http://www.eclipse.org/modeling/gmf/.
[8] http://www.omg.org/spec/XMI/.

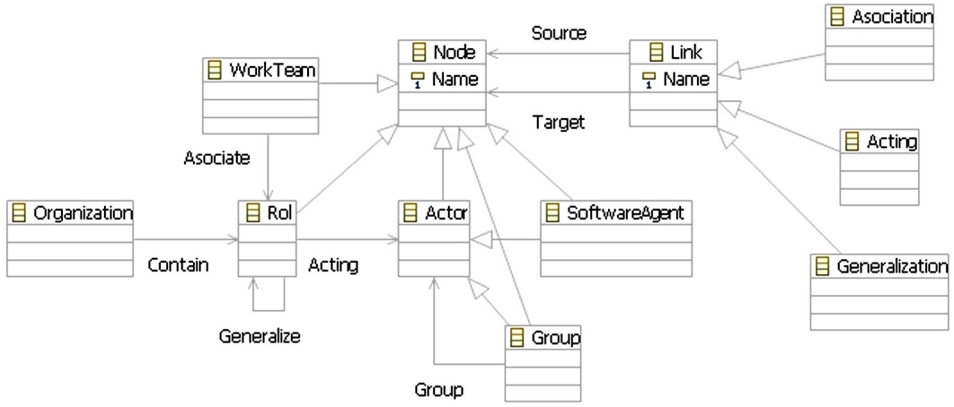

Fig. 7. *Ecore* model that defines the abstract syntax of the *Sociogram* models.

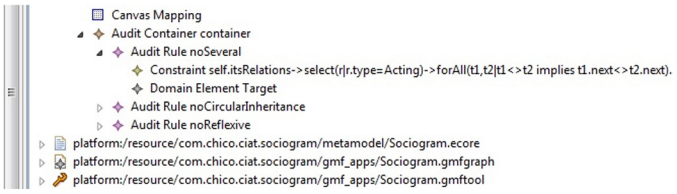

Fig. 8. Definition of an OCL constraint in the *domain model*.

with regard to the ease of use, usefulness and intention to use of the models,
from the point of view of interactive groupware analysts
within the context of the design of interactive groupware applications.

Thus, our **research questions** can be: $RQ_{\#1}$:"*Is CIAN perceived as easy to use and useful for interactive groupware requirements modeling?*" and $RQ_{\#2}$:"*Is there an intention to use CIAN in the future for designing interactive groupware systems?*". We base this empirical study on the *user perception* with regard to the quality of CIAN notation. We consider some of the dimensions of quality for requirements modeling methods proposed in (Abrahao et al., 2011). In order to answer these questions, two experiments were designed and conducted. In both the same hypotheses, experimental design and materials were used.

The **subjects** in the first experiment were 59 undergraduate students enrolled in the fourth and fifth years of Computer Science Degree at the University of Castilla-La Mancha in Ciudad Real. This experiment was organized as a compulsory part of the course. The subjects could obtain extra punctuation in the final mark of the subject, but only in the case that the modeling activities have a certain quality and were passed. Regarding the prior knowledge of respondents, 100% of them had completed, or were taking at the time of the experiment, the *Human–Computer Interaction* subject, and thus had both theoretical and practical knowledge about *modeling task*. In particular, 96.61% had some previous knowledge about and done practical activities in modeling tasks with the CTT

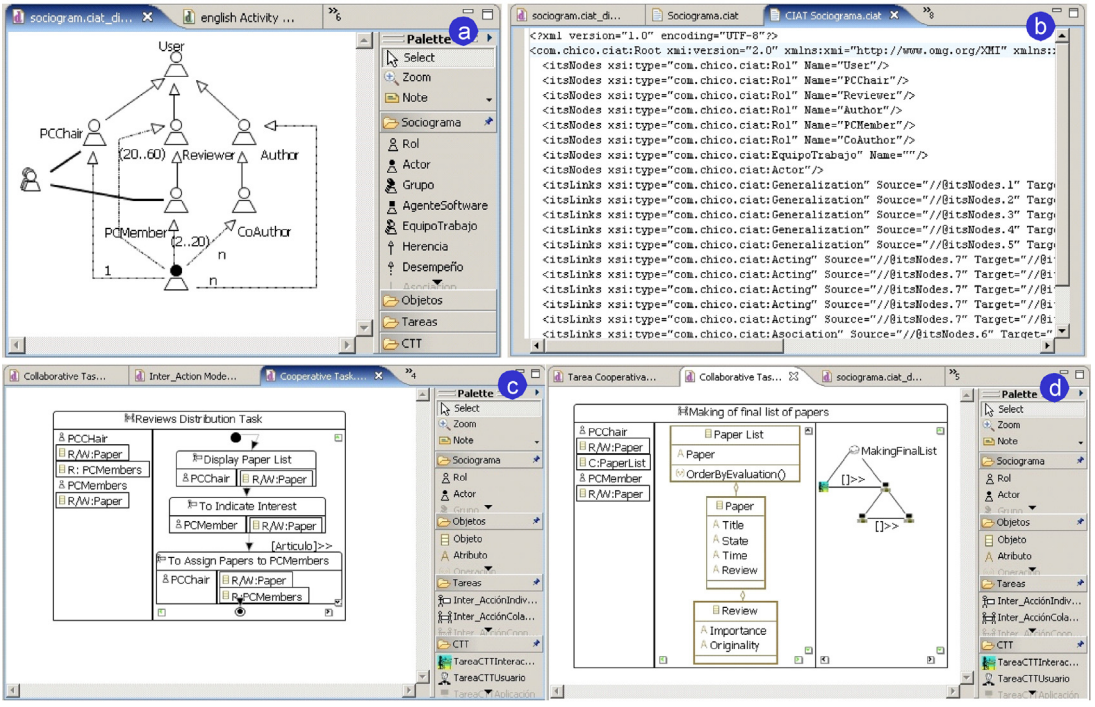

Fig. 9. (a) *Sociogram* edition using the CIAT tool (*left*). (b) XMI code generated associated with the model (*right*). (c) Modeling of a cooperative task using CIAT. (d) Modeling of a collaborative task using CIAT.

Table 6
Valuations of theoretical knowledge in modelling (with UML and CTT) and CSCW.

Areas	Undergraduate students		Professionals	
	Mean	Std dev	Mean	Std dev
Modeling of software systems using UML notation	3.12	0.95	2.63	0.81
Modeling of single-user systems using CTT notation	2.95	0.97	2.75	0.77
Modeling of cooperative systems using CTT notation	2.58	1.07	2.00	0.63
Computer Supported Cooperative Work (CSCW) systems	1.61	1.31	2.31	1.01

QuestionB7. Value from 1 to 5 your level of **THEORETICAL KNOWLEDGE** in the following areas (1 = very low, 5 = very high).

notation. In relation with their knowledge about the UML notation, 12 out of 59 participants had not studied *Software Engineering*, but the rest had some previous knowledge about this notation. As for their knowledge about collaborative systems, 30.51% of the 59 respondents had completed or were studying the *Groupware Engineering* subject when they took part in the study, compared to 69.49% that did not attend. But 100% of participants did not have any previous experience or knowledge on modeling using the CIAN notation.

In order to contrast the results of the first experiment a replica was conducted. In this case the participants were a set of 16 professionals that voluntarily took part in the experiment. These subjects were recruited in an introductory course in *Human–Computer Interaction* contents given by the Department of Information Systems and Technologies at the University of Castilla-La Mancha (in Ciudad Real). The medium age of the participants was 32 (between 24 and 41 years old). They had between 1 and 17 years of working experience in software development. In the replica, before running the experiment, some preliminary seminars about task modeling and collaborative systems were given. Although in this case the participation was voluntary and has not repercussion for them (as occurred in the first one, in which the motivation was extrinsic), the experimenters observed that they were very motivated to perform well in the study (intrinsic motivation) and very interested in the topic.

In order to learn more about the *profile* of the participants, they were asked to rate their knowledge, both theoretical and practical, in some of the areas. In Tables 6 and 7 we can see the scores on these aspects.

The subjects were also asked about their general perception of the *use of conceptual models*. 88.14% of students considered justifiable the effort made for modeling software systems, in general, and groupware systems, in particular. In the case of professionals 72.5% of respondents considered useful the use of conceptual models. Although the percentage in both cases is high, students have a better impression about the use of conceptual models. This perception is related to the different background and experience of both types of participants. Some professionals considered the use of conceptual specifications a desirable practice and appreciate their use, but pointed out that in professional contexts sometimes these specifications techniques are not used in an suitable way and the effort performed in their creation is not profitable.

Following, in the two experiments, a **training session** was developed to provide the participants with the necessary knowledge to do the tasks required in the experiment. During this learning session they were given some documentation with the aim to provide them with the information needed to learn about CIAN and to carry out the experimental tasks. This documentation was planned to be read in about 25 min on average and it was composed of the following parts: (a) *An introduction to the CIAN notation* in which the objectives of the notation were explained. The reason for including this part was to balance the knowledge needed by all the participants to carry out the experimental tasks; (b) *Examples of use of the CIAN notation*: A solved example of use of the CIAN notation and all the diagrams to be performed using it, and (c) *Instructions* to follow to carry out the tasks in the experiment. Next the subjects performed several **practical activities**, concerning the creation of models using the CIAN notation, some analysis and understandability exercises of CIAN models, the comparison with CTT models for the same requirements and, finally, some activities about modifying models.

Once the aforementioned exercises had been completed by the subjects, a **qualitative questionnaire** was provided. Each subject was required to answer some questions related to the *ease of use*, the perceived *complexity* of each model, as well as the *suitability* and *usefulness* of the notation and specific aspects related to some of the CIAN diagrams (the *process model* and the differentiate modeling of cooperative and collaborative tasks). Also, some questions about *intention to use* were provided.

5.2. Data analysis and interpretation

In the two experiments, once the task had been carried out, we collected the qualitative questionnaires filled in by the participants and carried out a descriptive analysis of the data. The main results of the experiments are discussed in the next subsections.

5.2.1. Evaluation of CIAN diagrams

In these two empirical studies we measure the user perception with regard to the quality of the notation. For this, we measured some *aspects* (Abrahao et al., 2011) as the *perceived ease of use*, the *perceived usefulness* and the *intention to use*. The *perceived ease of use* is the degree to which a person believes that using a particular method would be effort-free. This variable represents a perceptual judgment of the effort required to use a method. The *perceived usefulness* expresses the degree to which a person believes that a particular method will achieve its intended objectives. This variable represents a perceptual judgment of the method's effectiveness. The *intention to use* is defined as the extent to which a person intends to use a particular method. This variable represents a perceptual judgment of the efficacy of the method – that is, whether it is cost-effective. All these aspects were measured by means of the subjective *valuation* provided by the participants in the experiment.

Table 7
Valuations of practical knowledge in modeling (with UML and CTT).

Areas	Undergraduate students		Professionals	
	Mean	Std dev	Mean	Std dev
Modeling of software systems using UML notation	3.03	1.03	2.44	1.21
Modeling of single-user systems using CTT notation	3.05	0.97	2.50	0.73
Modeling of cooperative systems using CTT notation	2.59	1.08	1.94	0.68

QuestionB8. Value from 1 to 5 your level of **PRACTICAL KNOWLEDGE** in the following areas (1 = very low, 5 = very high).

Table 8
Valuations of *ease of use* of each of the diagrams proposed by CIAN.

Graphical specification technique	Undergraduate students		Professionals	
	Mean	Std dev	Mean	Std dev
Sociogram	2.75	1.33	1.56	0.89
Responsibilities models	1.78	1.23	2.06	1.06
Participation table	1.63	1.00	1.56	0.63
Process model	2.53	0.82	2.94	1.06
Cooperative task models	2.86	0.92	2.75	1.06
Collaborative task models	3.19	0.96	3.10	1.05
Interaction models (CTT)	2.68	1.18	2.15	0.65

QuestionNot2. Mark with a value from 1 to 5 the *ease of use* of the following diagrams (1 = *very easy to use*, 5 = *very difficult to use*).

Firstly, the participants were asked about their *preference* regarding the use of textual or graphical representations for the specification of software systems. 85.75% of the students preferred the graphical notations, compared to 11.86% that preferred textual notations. The remaining 3.39% did not choose any of these options. In the case of professionals 81.3% of the subjects preferred graphical representations, compared to 18.8% that preferred textual specifications. Regarding this aspect, the percentages are very similar.

The *perceived ease of use* was measured via the judgment of the participants about how easy or difficult they found it to use the several diagrams proposed in the CIAN notation according to five linguistic labels. The subjects rated each diagram on a scale from 1 (*very easy to use*) to 5 (*very difficult to use*) in accordance with their perception of the ease of use of the diagrams and their experience during the realization of the modeling exercises. Table 8 shows the mean of the scores assigned by the participants (students and professionals) to each diagram in CIAN notation.

As Table 8 shows, the modeling of collaborative tasks was considered as the most complex diagram by both groups of subjects. In particular, regarding the modeling of these tasks, 55.93% of the students and 68.75% of professionals considered that it was not easy to extract the *interaction model* (in CTT notation) from the definition of *shared context*.

As for the modeling language as a whole, some aspects related to the *perceived complexity* were also evaluated. The participants were asked about the *number of diagrams* to be created, as well as the *number of icons* that make up the notation. In relation to the number of diagrams, 71.19% of the students and 62.5% of professionals considered the number of models to create high. However, in relation to the number of icons, the vast majority (73.2% of the students and 81.25% of professionals) thought that the number of icons was adequate.

The participants also judged how *simple* or *easy to follow* this method for requirements modeling is. 72.88% of the students and 81.25% of professionals expressed that they had no problems when following the modeling method.

Also a set of adjectives were provided to users so that they could describe their perception of the CIAN notation as a whole. 79.66% of the students considered the CIAN notation a *complete* one. In the case of professionals the percentage was very similar (78.56%). Also the *homogeneity* of the notation was assessed positively by 81.36% of the students and 87.5% of professionals.

Regarding the *usefulness* of the CIAN notation, 76.27% of the students rated the notation as *useful*, compared to 18.64% that did not consider it, while the remaining 5.08% did not make comments about this issue. The percentage of professionals that considered useful the notation was of 70.25%. This result is consistent with the answer data by this group of respondents in relation to their general perception of the *use of conceptual models.*

Finally, the participants answered a question about their *intention to use* the proposed modeling method. In the case of students 76.27% indicated their intention to use this requirements modeling method in the future, compared to 22.03% that answered negatively. The remaining 1.69% did not comment. In the case of professionals 70.3% indicated their intention to use this specification technique. Again, the given answer to this question is consistent with the subjective perception of this group about usefulness of conceptual models, in general, and CIAN, in particular.

5.2.2. Evaluation of specific models

The questionnaire also included questions related to the main diagrams (related to the specification of group work) proposed by the notation. One of these diagrams is the *process model*. In relation to the information specified in this kind of representation, 72.88% of the students considered it useful to graphically display the *task-role-object* relationship. 5.08% did not consider it useful, while the remaining 22.03% considered it indifferent. In the case of professionals 93.75% of the respondents considered useful to visualize in this diagram the relationship of these three elements, while the 6.25% considered it indifferent. In this case, the difference in the answer given by two groups is very significant. Professional appreciated specially the capacity of CIAN for representing jointly these three aspects.

In relation to the type of representation more adequate to represent the *work structure* into various levels of abstraction, in the literature we find two main techniques: representations in tree form (such as that used in CTT) or graph representations (such as used in the *process* diagram in CIAN). The subjects were asked about this issue. In the case of students 35.59% considered most appropriate representation in tree form, compared to 49.15% that felt more appropriate to use graphs for this purpose. The remaining 15.25% considered it irrelevant. In the case of professionals the percentages are inverted. 43.74% considered more appropriate a tree-based representation, compared to 31.25% that preferred graph-based specifications. The remaining 25% considered this issue as irrelevant.

On the distinction between cooperative and collaborative tasks, 84.75% of students found it necessary to model cooperative and collaborative tasks in a differentiated way, compared to 93.75% of professionals that appreciate the differentiate specifications of these two kinds of tasks. In relation to the ability of CIAN to clearly express the difference between both types of tasks, 86.44% of students and 93.75% of professionals considered it adequate. The vast majority of respondents (79.66% of students and 87.5% of professionals) considered it necessary to model the *shared context* in collaborative tasks, and a greater number (82.03% of students and 93.75% of professionals) considered it appropriate to make the distinction between the different *visualization areas* provided by CIAN (*collaborative visualization area*, *individual visualization area* and the *exclusive edition area*). The differentiate modeling of cooperation/collaboration, the inclusion of shared context specification and the identification of visualization areas are three of the most important and distinctive contributions of CIAN notation. These three aspects have been very much appreciated by both groups, and especially by professionals.

The subjects were also asked about the use of CTT notation in the *interaction modeling* stage. 57.63% of the students considered that the use of this notation affected the uniformity of CIAN notation negatively. In the case of professionals this percentage increases up to 75%.

5.2.3. Final open questions

Finally, we included a set of open questions to collect the opinion of subjects about the strengths and weaknesses of the proposal, and to gather suggestions for improvement. They were also asked about the use of CIAN versus other widespread notations in the area of modeling in HCI (as CTT).

In general, the respondents considered as the outstanding points of CIAN its *expressiveness*, as well as the *easy to understand* the models specified within this notation. Among all the diagrams, the *process model* was considered the most useful to specify the group work (especially by professionals). The capacity for representing work flow, iterations and temporal restrictions were considered as very useful. Professionals also appreciated the use of CIAN as a communication tool to be used in meetings with clients, especially in requirements specification phase.

As for the weaknesses, some subjects noted that the process of creating the diagrams was somewhat laborious, because they considered the number of models high. They suggested as a possible solution that the tool should allow the automatic generation of some representation (if not entirely, at least partially) from others. Some subjects indicated that the information shown in some diagrams was redundant and proposed the elimination of some representations. Moreover, and as noted above, modeling of collaborative tasks was the one that caused major difficulties for designers. Some professionals pointed out the learning curve and the relation effort-benefits of applying this method in real contexts as two of the most questionable aspects of the method.

Regarding the improvement proposals, some of the subjects suggested reducing the number of diagrams and eliminating the textual representations. Also, some of them proposed replacing CTT in the last stage and creating a diagram similar to the *process model* to specify the interaction.

As for the comparison of CIAN and CTT models, the participants felt that, although the modeling with CIAN is more laborious than with CTT, the former is a more complete proposal, and it allows specifying many aspects that CTT does not include (notifications, temporal constraints, tasks that involve several roles, objects manipulated, etc.). On the other hand, they considered that using graph diagrams was more manageable and maintainable than a representation in tree form (such as the one used in CTT).

6. Conclusions and future works

Building software, and particularly *groupware*, is a complex task. In order to make it easier, one of the paradigms that is gaining ground is *Model-Driven Engineering*. The MDE paradigm is based on the use of models. Reviewing the approaches that deal with modeling of user interfaces supporting collaborative tasks, we have detected that there is no proposal linking both interactive and collaborative characteristics. The aim of the work presented in this paper was to propose a graphical language for the design of collaborative and interactive issues in groupware systems, in this way, solving this lack. With the aim of creating a coherent set of modeling elements, we based them on a conceptual framework that was materialized in a metamodel. In the definition of the conceptual framework we considered the strengths of several conceptual frameworks and modeling languages existing in the literature. The goal was that the notation proposed could give more complete support and greater semantics than the rest of proposals that deal within the design of the presentation layer in groupware systems.

The set of concepts that form the conceptual framework were divided into four analysis (and modeling) views or perspectives for specifying interactive *groupware* systems: the *organizational view* (organizational structure), the *process view* (work structure and work flow), the *data view* (information manipulated) and the *interaction view* (interaction with the presentation layer). All these views are related between them, giving rise to an overall schema that relates all of these perspectives.

The metamodel created was not only used as the base for the definition of a DSL for modeling such systems (which we called CIAN), but also thanks to its metamodel foundation, this notation includes clear syntactic and semantic definitions. Additionally, using the tools provided by the *Eclipse Foundation* in its metamodeling project (*Eclipse Modeling Project*), we have obtained the tools that support modeling with CIAN. Using GMF and EMF it is possible to automate the generation of a graphical editor to create and manipulate models of a certain domain specified by means of a meta-model. This also helps to demonstrate that our language approach is computable.

Finally, we conducted two empirical studies of the use of the CIAN notation. In the first one the subjects were undergraduate students, while in the replication a group of 16 professionals in software development were involved. We proposed two research questions related to the ease of use, the usefulness and the intention to use of CIAN. In relation to $RQ_{\#1}$: "*Is CIAN perceived as easy to use and useful for interactive groupware requirements modeling?*" the majority of the participants found the CIAN notation quite useful and easy to use in performing the specification of interactive groupware requirements. In relation to $RQ_{\#2}$: "*Is there an intention to use CIAN in the future for designing interactive groupware systems?*" the majority of the respondents were positive about the use of the CIAN notation in the future. The results provided in these experimental studies denote positive perceptions about the proposed DSL and provided as a preliminary feedback with regard to the quality of the concrete syntax of CIAN. With these results, we can conclude that CIAN is a notation that has proven useful for helping software engineers in the modeling process of interactive and collaborative aspects in groupware applications.

Following this line of work, we plan to consider some of the suggestions made by the respondents in the study. In particular we want to improve the support provided by CIAT for supporting trazability between diagrams and specifications. Also, and within future work, one important effort will be the performing of an experiment to validate the *usability* and *level of acceptance* of the CIAT editor. We also intend to replicate the experiments conducted to measure other quality attributes of graphical notations, such as the *maintainability* of the proposed language. Finally, another aspect that we consider in the future will be to enhance the metamodel including concepts related with *awareness* (Dourish and Bellotti, 1992) support in collaborative user interfaces.

Acknowledgments

This work has been partially supported by the INTEGroup, GITE-LEARN e iColab projects, funded by the Junta de Comunidades de Castilla-La Mancha (Spain). We thank Pedro Pablo Sánchez Villalón for his selfless help in reviewing the English language and the students and professionals who participated in the two empirical studies.

References

Abrahao, S., Insfran, E., Carsí, J.A., Genero, M., 2011. Evaluating requirements modeling methods based on user perceptions: a family of experiments. Information Sciences 181 (2011), 3356–3378.

Antunes, P., Guimaraes, N., Nunes, R., 1991. Extending the user interface to the multiuser environment. In: European Computer Supported Cooperative Work (ECSCW'91), CSCW Developers Workshop, Amsterdam.

Antunes, P., Guimaraes, N., 1994. Multiuser Interface Design in CSCW Systems.

Bezivin, J., Jouault, F., Touzet, D., 2005. Principles, standards and tools for model engineering. In: Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'2005), pp. 28–29.

Bolognesi, T., Brinskma, H., 1987. Introduction to the ISO specification language LOTOS. Computer Networks and ISDN Systems 14, 25–59.

Carlsen, S., 1998. Action port model: a mixed paradigm conceptual workflow modeling language. In: Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems, IEEE Computer Society.

Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., Seguin, C., 1998. Java object-sharing in HABANERO. Communications of the ACM 41 (June (6)).

Cortes, M., Mishra, P., 1996. DCWPL: a programming language for describing collaborative work. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 21–29.

Cortes, M., 2000. Coordination language for building collaborative applications. Computer Supported Cooperative Work: CSCW: An International Journal 9 (1), 5–31.
Cox, K., Walker, D., 1993. User Interface Design. Prentice Hall.
Clayberg, E., Rubel, D., 2006. Eclipse: Building Commercial-Quality Plug-ins, 2nd ed. Addison Wesley Professional.
Crowston, K., Rubleske, J., Howison, J., 2004. Coordination theory. In: Sharpe, M.E., Zhang, P., Galletta, D. (Eds.), Human–Computer Interaction in Management Information Systems, vol. I.
David, B., Delotte, O., Chalon, R., Tarpin-Bernard, F., Saikali, K., 2003. Patterns in collaborative system design, development and use. In: 2nd Workshop on Software and Usability Cross-Pollination: The Role of Usability Patterns.
de Cesare, S., Lycett, M., 2002. Business Modelling with UML: Distilling Directions for Future Research, Information Systems Analysis and Specification.
de Cesare, S., Serrano, A., 2006. Collaborative modeling using UML and business process simulation. In: IEEE Proceedings of the 39th Hawaii International Conference on System Sciences 2006 (HICSS'06).
Diaper, D., Stanton, N.A., 2004. The Handbook of Task Analysis for Human–Computer Interaction. Lawrence Erlbaum Associates, Publishers, Mahwah, NJ.
Dillenbourg, P., Baker, M., Blaye, A., O'Malley, C., 1996. The evolution of research on collaborative learning. In: Spada, E., Reiman, P. (Eds.), Learning in Humans Machine: Towards an Interdisciplinary Learning Science. Elsevier, Oxford, pp. 189–211.
Dijkman, R.M., Quartel, D.A.C., Ferreira Pires, L., van Sinderen, M.J., 2003. An approach to relate viewpoints and modeling languages. In: Proceedings of the 7th IEEE Enterprise Distributed Object Computing (EDOC) Conference, Brisbane, Australia.
Dix, A., 1993. Human–Computer Interaction. Prentice-Hall, Englewood Cliffs, NJ.
Dourish, P., Bellotti, V., 1992. Awareness and coordination in shared workspaces. In: Proceedings of the Conference on Computer Supported Cooperative Work CSCW'92, Toronto, Canada. ACM Press, New York.
Ellis, C.A., Gibbs, S.J., Rein, G.L., 1991. Groupware: some issues and experiences. Communications of the ACM 34 (1), 39–58.
Eriksson, H.E., Penker, M., 2000. Business Modeling with UML. Business Patterns at Work.
Favre, J.M., 2005. Foundations of meta-pyramids: languages vs. metamodels, episode II: story of Thotus the Baboon1, presented at Dagsthul Seminar on Model Driven Approaches for Language Engineering.
France, R., Rumpe, B., 2007. Model-driven development of complex software: a research roadmap. In: IEEE Computer Society, 2007 Future of Software Engineering, pp. 37–54.
Feilkas, M., 2006. How to represent models, languages and transformations? In: Proceedings of the 6th OOPSLA Workshop on Domain-specific Modeling (DSM'06), pp. 204–213.
Fenton, N., Pfleeger, S., 1997. Software Metrics: A Rigorous Approach, 2nd ed. Chapman & Hall, London.
Frankel, D.S., 2004. Domain-specific modeling and model driven architecture. MDA Journal.
Garrido, J.L., 2003. AMENITIES: Una metodología para el desarrollo de sistemas cooperativos basada en modelos de comportamiento y tareas. Departamento de Lenguajes y Sistemas Informáticos. Granada, PhD Thesis, Universidad de Granada.
Garrido, J.L., Gea, M., 2001. Modelling Dynamic Group Behaviours. Design Specification and Verification of Interactive Systems DSVIS 2001. Springer-Verlag.
Garrido, J.L., Gea, M., Rodríguez, M.L., 2005. Requirements Engineering in Cooperative Systems. Requirements Engineering for Sociotechnical Systems. Idea Group Inc., USA, pp. 226–244.
Genero, M., Piattini, M., Calero, C. (Eds.), 2005. Metrics for Software Conceptual Models. Imperial College Press, United Kingdom.
Gemino, A., Wand, Y., 2005. Complexity and clarity in conceptual modeling: comparison of mandatory and optional properties. Data and Knowledge Engineering 55, 301–326.
Guerrero, L.A., Fuller, D.A., 1999. Design patterns for collaborative systems. In: Proceedings of the Fifth International Workshop on Groupware (CRIWG).
Guizzardi, G., 2005. Ontological Foundations for Structural Conceptual Models. Centre for Telematics and Information Technology, University of Twente, The Netherlands.
Guizzardi, G., Ferreira Pires, L., van Sinderen, M.J., 2002. On the role of domain ontologies in the design of domain specific visual modeling languages. In: OOPSLADSVL.
Guizzardi, G., Wagner, G., Guarino, N., van Sinderen, M., 2004. An ontologically well-founded profile for UML conceptual models. In: CAiSE 2004.
Grudin, J., 1994. Groupware and social dynamics: eight challenges for developers. Communications of the ACM 37 (1), 92–105.
Harstson, R., Gray, P., 1992. Temporal aspects of tasks in the user action notation. Human Computer Interaction 7, 1–45.
Henderson-Sellers, B., 2007. On the challenges of correctly using metamodels in method engineering. In: Fujita, H., Pisanelli, D. (Eds.), Keynote Paper in New Trends in Software Methodologies, Tools and Techniques. Proceedings of the sixth SoMeT'07. IOS Press, pp. 3–35.
Henderson-Sellers, B., 2011. Bridging metamodels and ontologies in software engineering. Journal of Systems and Software 84, 301–313.
Hummes, J., Merialdo, B., 2000. Design of extensible component-based groupware. Computer Supported Cooperative Work: CSCW: An International Journal 9 (1), 53–74.
Isla, J.L., Gutiérrez, F.L., Gea, M., 2006. Supporting social organization modelling in cooperative work using patterns. In: Shen, W., et al. (Eds.), Computer Supported Cooperative Work in Design, vol. 3865. LNCS, Springer, pp. 112–121.
ISO/IEC, 2001. ISO/IEC 9126-1: Software Engineering-Software Product Quality—Part 1: Quality Model. International Organization for Standardization, Geneva, Switzerland.
Johansen, R., 1988. Groupware: Computer Support for Business Teams. The Free Press, New York.
Johnson, P., 2004. Interactions, collaborations and breakdowns. In: 3th Task Models and Diagrams for User Interface Design (TAMODIA 2004), Prague, Czech Republic.
Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., Volkel, S., 2009. Design guidelines for domain specific languages. In: Proceedings of the 9th OOPSLA Workshop on Domain-specific Modeling, pp. 7–13.
Katzenbach, J.R., Smith, D.K., 1999. The Wisdom of Teams: Creating the High-Performance Organization. Harper Business, New York.
Kelly, S., Pohjonen, R., 2009. Worst practices for domain-specific modeling. IEEE Software 26 (4), 22–29.
Koch, M., 1996. Design issues and model for a distributed multi-user editor. Computer Supported Cooperative Work: An International Journal 3 (3–4), 359–378.
Kolovos, D.S., Paige, R.F., Kelly, T., Polack, F.A.C., 2006. Requirements for domain-specific languages. In: First ECOOP Workshop on Domain-specific Program Development (ECOOP'06).
Kurtev, I., Bézivin, J., Jouault, F., Valduriez, P., 2006. Model-based DSL frameworks. In: OOPSLA Companion, pp. 602–616.
Kuutti, K., 1991. The concept of activity as a basic unit of analysis for CSCW research. In: Second European Conference on Computer Supported Cooperative Work (ECSCW'91).
Lacaze, X., Palanque, P., 2004. Comprehensive handling of temporal issues in tasks models: what is needed and how to support it? In: Workshop the Temporal Aspects of Work for HCI (CHI 2004), Vienna, Austria.
Li, X., 2005. Using UML in conceptual modeling: towards an ontological core. In: Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE05), Porto, Portugal.
Lim, Y.K., 2004. Multiple aspect based task analysis (MABTA) for user requirements gathering in highly-contextualized interactive system design. In: Proceedings of the 3rd Annual Conference on Task Models and Diagrams (TAMODIA 2004), ACM International Conference Proceeding Series, Prague, Czech Republic.
Limbourg, Q., 2004. Multi-Path Development of User Interfaces, PhD Thesis, Université catholique de Louvain, p. 291.
Limbourg, Q., Vanderdonckt, J., 2009. Multi-path transformational development of user interfaces with graph transformations. In: Seffah, A., Jean Vanderdonckt, M., Desmarais (Eds.), Human-Centered Software Engineering, HCI Series. Springer, London, pp. 109–140.
Lukosch, S., Schümmer, T., 2006. Groupware development support with technology patterns. International Journal of Human Computer Studies 64 (7), 599–610.
Luyten, K., 2004. Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development, PhD Thesis, Universiteit Limburg.
Maes, A., Poels, G., 2007. Evaluating quality of conceptual modelling scripts based on user perceptions. Data and knowledge Engineering 63 (2007), 701–724.
Malone, R.W., Crowston, K., 1990. What is coordination theory and how can it help design cooperative work systems? In: ACM Conference on Computer Supported Cooperative Work.
Markopoulos, P., Marijnissen, P., 2000. UML as a representation for interaction design. In: OZCHI.
Marsic, I., 2001. An architecture for heterogeneous groupware applications. In: Proceedings of the International Conference on Software Engineering, pp. 475–484.
Mayer, R.E., 1989. Models for understanding. Review of Educational Research 59 (1), 43–64.
Miao, Y., Haake, J.M., 1998. Supporting concurrent design by integrating information sharing and activity synchronization. In: Proceedings of the 5th ISPE International Conference on Concurrent Engineering Research and Applications (CE98), Tokyo, Japan.
Molina, A.I., Redondo, M.A., Ortega, M., 2006. Applying pattern-based techniques to design groupware applications. In: Proceedings of the Third International Conference on Cooperative Design, Visualization and Engineering (CDVE2006), Mallorca, Spain. LNCS, Springer-Verlag.
Molina, A.I., Redondo, M.A., Ortega, M., 2009a. A review of notations for conceptual modeling of groupware systems. In: Macías, J.A., Granollers, T., Latorre, P.M. (Eds.), New Trends on Human–Computer Interaction: Research, Development, New Tools and Methods. Springer-Verlag, ISBN 978-1-84882-351-8, pp. 75–86.
Molina, A.I., Redondo, M.A., Ortega, M., 2009b. A methodological approach for user interface development of collaborative applications: a case study. Science of Computer Programming 74, 754–776.
Moody, D.L., 2005. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data and Knowledge Engineering 55 (3), 243–276.
Moody, D.L., 2009. The "Physics" of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering 35 (6).
Moore, B., Dean, D., Gerber, A., Wagenknecht, G., Vanderheyden, P., 2004. Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework.

Mori, G., Paternò, F., Santoro, C., 2004. Design and development of multidevice user interfaces through multiple logical descriptions. IEEE Transactions on Software Engineering (August), 507–520.
Nelson, J., Poels, g., Genero, M., Piattini, M., 2005. Quality in conceptual modeling—five examples of the state of art. Data and Knowledge Engineering 55 (3), 237–242.
Nelson, H.J., Poels, G., Genero, M., Piattini, M., 2012. A conceptual modeling quality framework. Software Quality Journal 20, 201–228.
OMG, 2003. OCL 2.0—OMG Final Adopted Specification. Object Management Group.
Pankoke-Babatz, U., 1994. Reflections on concepts of space and time in CSCW. In: Proceedings of the ECCE 7: Seventh European Conference on Cognitive Ergonomics. Human Computer Interactions: From Individuals to Groups, GMD Studien, Sankt Augustin, Germany.
Paternò, F., 1999. Model-Based Design and Evaluation on Interactive Applications. Springer Verlag.
Paternò, F., 2001a. Towards a UML for interactive systems. In: Proceedings of the 8th International Conference on Engineering for Human–Computer Interaction, Lectures Notes Computer Science.
Paternò, F., 2001b. Task models in interactive software systems. In: Chang, S.K. (Ed.), Handbook of Software Engineering & Knowledge Engineering. World Scientific Publishing Co.
Paternò, F., 2004. ConcurTaskTrees: an engineered notation for task models. In: Diaper, D., Stanton, N.A. (Eds.), The Handbook of Task Analysis for HCI. LEA, Mahwah, NJ, pp. 483–501.
Paternò, F., Santoro, C., 2002. One model, many interfaces. In: Proceedings of Computer-Aided Design of User Interfaces (CADUI 2002), Valenciennes, France.
Pelechano, V., Albert, M., Javier, M., Carlos, C., 2006. Building tools for model driven development comparing Microsoft DSL tools and eclipse modeling plug-ins. In: Desarrollo de Software Dirigido por Modelos—DSDM'06.
Penichet, V.M.R., Lozano, M.D., Gallud, J.A., Tesoriero, R., 2007. Task modelling for collaborative systems. In: Proceedings of the 6th International Workshop on TAsk MOdels and DIAgrams: TAMODIA.
Penichet, V.M.R., Lozado, M.D., Gallud, J.A., Tesoriero, R., 2010. Requirement-based approach for groupware environments design. Journal of Systems and Software 83, 1478–1488.
Pinelle. D., 2004. Improving Groupware Design for Loosely Coupled Groups. Department of Computer Science, PhD Thesis, University of Saskatchewan.
Preece, J., Rogers, Y., Sharp, H., 2011. Interaction Design: Beyond Human–Computer Interaction. John Wiley & Sons.
Pribeanu, C., 2002. Investigating the relationship between the task model and the design model in a task-based approach to user interface design. In: Proceedings of the 1st International Workshop on Task Models and Diagrams for User Interface Design – Tamodia 2002, Bucarest.
Pribeanu, C., Vanderdonckt, J., 2002. Exploring design heuristics for user interface derivation from task and domain models. In: Computer-Aided Design of User Interfaces (CADUI 2002).
Redondo, M.A., 2005. Análisis de accciones e inter-acciones en sistemas colaborativos. In: Redondo, M.A., Bravo, C., Lorés, J. (Eds.), Presente y Futuro de la Docencia e Investigación en Interacción Persona-Ordenador. , pp. 273–283.
Roth, J., Unger, C., 1998. DreamTeam—a synchronous CSCW environment for distance education. In: ED-MEDIA/EDTELECOM'98, Freiburg, Germany.
Rubart, J., Dawabi, P., 2004. Shared data modeling with UML-G. International Journal of Computer Applications in Technology 19.
Rumbaugh, J., Jacobson, I., Booch, G., 2000. El Lenguaje Unificado de Modelado. Manual de Referencia. Addison-Wesley, Madrid.
Saeki, M., 2003. Embedding metrics into information system development methods: an application of method engineering technique. Lecture Notes in Computer Science 2681, 374–389.
Sarin, S., Greif, I., 1985. Computer-based real-time conferencing systems. IEEE Computer 18 (10.).
Shneiderman, B., Plaisant, P., 2010. Designing the user interface. Strategies for Effective Human–Computer Interaction. Pearson.
Slagter, R., ter Doest, H., 2001. The CoCoWare Component Architecture. GigaCSCW deliverable (D2.1.4).
Schulte, R., 1995. Three-tier computing architectures and beyond, Published Report Note R-401-134, Gartner Group.
Sousa, K.S., Mendoza, H., Vanderdonckt, J., 2007. Towards method engineering of model-driven user interface development. Task Model and Diagrams for User Interface Design, TAMODIA'2007, Toulouse, 7–9 November 2007, Springer-Verlag, Berlin. Lecture Notes in Computer Science 4849, 112–125.
Stefik, M., Bobrow, D.G., Foster, G., Lanning, S., Tatar, D., 1987. WYSIWIS revised: early experiences with multiuser interfaces. ACM Transactions on Office Information Systems 5 (2), 147–167.
Stocq, J., Vanderdonckt, J., 2004. A domain model-driven approach for producing user interfaces to multi-platform information systems. In: Proceedings of the Working Conference on Advanced Visual Interfaces, AVI 2004, Gallipoli, Italy. ACM Press.
Szekely, P., 1996. Retrospective and challenges for model-based interface development. In: Proceedings of the 2nd International Workshop on Computer-Aided Design of User Interfaces CADUI'96, Presses Universitaires de Namur, Namur, pp. xxi–xliv.
Trætteberg, H., 2002. Model-based User Interface Design. Dept. of Computer and Information Sciences, PhD Thesis, Norwegian University of Science and Technology.
van der Aalst, W.M.P., Berthelmess, P., Ellis, C.A., Wainer, J., 2001. Proclets: a framework for lightweight interacting workflow processes. Journal of Cooperative Information Systems 10–14, 443–482.
van der Aalst, W.M.P., Kumar, A., 2001. A reference model for team-enabled workflow management systems. Data and Knowledge Engineering 38 (3), 335–363.
van der Veer, G.C., van Welie, M., 2000. Task based groupware design: putting theory into practice. In: Proceedings of DIS 2000, New York, USA.
van Welie, M., van der Veer, G.C., 2003. Groupware task analysis. In: Hollnagel, E. (Ed.), Handbook of Cognitive Task Design. LEA, New Jersey, pp. 447–476.
van Welie, M., van der Veer, G.C., Eliëns, A.,1998. An ontology for task world models. In: Design, Specification and Verification of Interactive Systems (DSV-IS'98). Springer.
Vanderdonckt, J., 2005. A MDA-compliant environment for developing user interfaces of information systems. In: Pastor, O., Falcão, J. (Eds.), CAISE 2005, LNCS 3520. Springer-Verlag, Berlin/Heidelberg, pp. 16–31.
Vanderdonckt, J., 2008. Model-driven engineering of user interfaces: promises, successes and failures. In: Buraga, S., Juvina, I. (Eds.), Proceedings of the 5th Annual Romanian Conf. on Human–Computer Interaction ROCHI'2008. Iasi, 18–19 September 2008. Matrix ROM, Bucarest, pp. 1–10.
Vanderdonckt, J., Berquin, P., 1999. Towards a very large model-based approach for user interface development. In: Paton, N.W., Griffiths, T. (Eds.), Proceedings of the 1st International Workshop on User Interfaces to Data Intensive Systems UIDIS'99. IEEE Computer Society Press, pp. 76–85.
Völter, M., 2009. MD* best practices. Journal of Object Technology 8 (6), 79–102.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation in Software Engineering—An Introduction. Kluwer Academic Publishers, Boston, MA.

Ana I. Molina, MSc and PhD in Computer Science, belongs to the *Computer Human Interaction and Collaboration* (CHICO) Research Group and works as Associate Professor at the University of Castilla-La Mancha (Spain). Her research interests are, among others, Model-Based User Interface Development, Human-Computer Interaction, Computer-Supported Cooperative Work (CSCW) and collaborative learning. She is author of numerous publications in several national and international conferences and journals.

Jesús Gallardo is Assistant Professor at the Department of Computer Science and Systems at the University of Zaragoza (Spain). His research interests include, among others, model-driven development of collaborative systems, awareness and collaboration protocols. He received his Ph.D degree in Computer Science in 2010 from University of Castilla-La Mancha.

Miguel A. Redondo has a PhD in Computer Science (2002) and is Associate Professor at the Department of Information Technologies and Systems at the University of Castilla – La Mancha (Spain). His research interests focuses on the fields of new Information Technologies applied to Education and Computer-Human Interaction. He is responsible of different research projects about development of mobile applications and groupware systems for learning programming.

Manuel Ortega is Full Professor at the Escuela Superior de Informática (College of Computer Science and Engineering) at the University of Castilla - La Mancha (Spain). He received his BSc (1982) and PhD (1990) degrees from Universidad Autónoma de Barcelona (Spain). He is the director of the *Computer Human Interaction and Collaboration* (CHICO) Research Group and is the Editor of the Iberoamerican Journal on Computers on Education IE Comunicaciones. His main interests are in the field of New Information Technologies applied to collaborative learning and Human-Computer Interaction.

William J. Giraldo is Associate Professor at the University of Quindío (Colombia). He received his PhD in Computer Science (2010) from the University of Castilla-La Mancha (Spain). His research interests include Human-Computer Interaction, Model-Driven Development and CSCW. As a result of this research, he is author of papers in several international conferences and journals.