*Journal of Universal Computer Science, vol. 14, no. 19 (2008), 3053-3070*
*submitted: 23/7/08, accepted: 29/10/08, appeared: 1/11/08 © J.UCS*

# Extending and Supporting Featured User Interface Models for the Development of Groupware Applications

**Víctor M.R. Penichet, María D. Lozano, José A. Gallud, Ricardo Tesoriero**
(Computer Systems Department, University of Castilla-La Mancha
02071 Albacete, Spain
victor.penichet@uclm.es, maria.lozano@uclm.es, jose.gallud@uclm.es
ricardo.tesoriero@uclm.es)

**María L. Rodríguez, José L. Garrido, Manuel Noguera, María V. Hurtado**
(Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada
E.T.S.I.I.T., c/Periodista Daniel Saucedo Aranda s/n,18071 Granada, Spain
mlra@ugr.es, jgarrido@ugr.es, mnoguera@ugr.es, mhurtado@ugr.es)

**Abstract:** This paper presents a proposal to tackle the design and development of user interfaces for groupware applications. This proposal includes important design and implementation issues of special relevance for this kind of interfaces. In particular, group awareness requirements in the development of groupware applications are addressed, both in the sense of the basic manipulation actions of the interface widgets, as well as in the sense of other kinds of group awareness in relation to the presence of actors, the roles they play in a concrete moment, etc. The design proposal we present is part of a complete development process (called TOUCHE) which defines a set of facets to describe Abstract Interaction Objects. These objects, at design level, provide the basis for the definition of Concrete Interaction Objects at implementation level within a software platform intended to facilitate the development of user interfaces for groupware applications. This way, we get an integral approach to tackle the development of this kind of user interfaces, taking into account in an explicit way the perception of the joint activity of a group of users involved in a common task and thus achieving a more effective collaboration.

**Keywords:** Groupware, group awareness, user interfaces
**Categories:** H.5.2, H.5.3

## 1 Introduction

Nowadays, groupware is considered a strategic tool for organizations. The user interface (UI) is a key factor that influences its degree of acceptance, since its functionalities facilitate capabilities for communication and collaboration between users working in a common task. There is no doubt that the user interface is a crucial aspect for the successful usability of this kind of applications, since it allows communication, collaboration and coordination activities among several users interacting with the system. When we face the development of a complex system to be used by different users simultaneously, the design of the user interface becomes even more important.

Therefore, methodologies and technologies related to user interface development should take into account the specific characteristics of groupware applications. Thus,

new challenges arise in relation to requirements, design and technology issues for groupware user interfaces.

Generally speaking, user interfaces of groupware applications should reflect group activity since they are managed by several users. The main issue to deal with is their complex behaviour due to their high level of activity and concurrence. The user interface of these systems should support the interaction between a user and the system so that individual tasks can be carried out, as well as the interaction between users through the system so that they can get involved in common tasks and enable the social interaction among the members of a group.

In addition, they should include information about which user is using the system, where they are working and what they are doing, i.e., appropriate group awareness mechanisms are required [Gutwin, 05]. In [Dourish, 92] group awareness is defined as the *understanding of the activities of others which provides a context for your own activity*. This context is used to ensure that individual contributions are relevant to the group activity as a whole, and to evaluate individual actions with respect to group goals and progress. Relevant information about the group itself allows their members to manage the collaborative work process properly.

In order to achieve an effective collaboration it is necessary to take into account some issues regarding group awareness [Schlichter, 98]:

- Informal: who are using the system, where they are located, etc.
- Group structure: Roles, responsibilities and status of the group members.
- Social: Knowledge about emotional state, attention and concern of users interested in collaboration issues.
- Workspace: Information about the interaction with other users in a shared workspace for an effective collaboration.

Also, it is important that the actions of a particular user can be shown to other users that collaborate in the same task (feedthrough) [Dix, 98; Gutwin, 04]. For example, the observation of other person that browses across the menu elements may give an indication about what one is doing or his (her) intentions. The Cameleon Reference Framework [Calvary, 03] is a reference model that can be used for comparing and reasoning about existing tools as well as for developing future run time infrastructures for distributed and plastic user interfaces. Moreover, that framework is the basis for user interfaces development in which user interaction is taken into account in an explicit manner.

At present, this framework has been successfully used and in this paper it is extended to allow the design of interfaces for groupware applications. Additionally, a specific platform for the design and implementation of groupware interfaces is introduced following the aforementioned reference framework.

The rest of this paper is organized as follows. Section 2 gives a background of related works. Section 3 describes a general process and the proposed extensions of conceptual models for the design of groupware user interfaces. Section 4 shows how the platform developed provides the necessary support in order to fulfil the previous featured UI abstract design for its successful implementation. In Section 5 several examples illustrating the proposal in relation to a particular case study are shown. Section 6 presents a brief discussion about the approach as a comparison with other previous ones. Finally, some conclusions and final remarks are presented in section 7.

## 2 Related Work

Several methods and techniques have been proposed to design user interfaces for computer systems. However, groupware applications have some particular features that need to be specially considered in order to achieve a better user interface design covering the users' real needs.

In the literature we can find different proposals for user interface design. Currently, most of them propose generic models at different abstraction levels (analysis and design stages) and afterwards they specify implementation details according to the available and selected platform, personalization characteristics, etc. All together sets the basis of Abstract Interaction Objects (AIOs). The first proposal based on this approach was TRIDENT (Tools foR an Interactive Development ENvironmenT) in 1990 [Bodart, 95]. TRIDENT is a set of tools that automatically generate the UI for highly-interactive applications. This approach uses AIOs during the UI design in order to get a selection of UI objects aside from implementation details. When the final platform and the concrete implementation features are decided in the implementation stage, the AIOs are translated into Concrete Interaction Objects (CIOs), depending on the specific platform where the software application, and therefore, its UI, is expected to run. These specific CIOs are also known as Widgets (Window Gadgets) or Controls or Physical Interactors (according to the IFIP terminology). The AIOs (Logical Interactors, according to the IFIP terminology) abstract features from the set of CIOs irrespectively of the final platform or environment. Vanderdonckt and Bodart [Bodart, 94] proposed a classification of six different types of AIOs that could be matched with zero, one or several CIOs depending on the environment. These CIOs can be simple or compound.

The IDEAS (Interface Development Environment within OASIS) methodology in 2001 [Lozano, 01] includes a Dialogue Model at the design stage to graphically represent the UI. This model would be the basis to get the final user interface at the implementation stage, because it is an abstract description of the actions, and their possible temporal relations, that the users may perform on the computer system through its user interface during an interactive session while carrying out certain tasks. Specifically, these descriptions are modelled by means of AIOs.

UMLi (The Unified Modeling Language for Interactive Applications) [da Silva, 00] appeared between 2000 and 2002 as a new notation extending UML to tackle the integrated design of computer applications and their corresponding user interfaces. It is also based on declarative models and aims to serve as a bridge between the UI developers and the application developers, i.e., the user interface is described jointly with the rest of the application. They propose a user interface diagram which includes six different types of AIOs or constructors that are used to specify the role played by each interactive object in a UI presentation model.

In 2004, more recent contributions consist in adding facets to describe the AIOs. Facets are a means to represent the nature –sometimes complex– of AIOs, and where each AIO may have several featured facets. In this sense, UsiXML (USer Interface eXtensible Markup Language) [Limbourg, 05b] is released as part of the work carried out by Limbourg and Vanderdonckt [Limbourg, 05a] at the Belgian Laboratory of Computer Human Interaction. UsiXML is a user interface description language that is independent of the final platform and other implementation details.

The Abstract User Interface Model (AUIM) depicts this situation based on AIOs and their relationships. All these concepts constitute a vocabulary independent of implementation concerns. A novelty introduced in [Limbourg, 05b] regarding AIOs is their distinction just between two types: AIOs that do not contain any other, called Abstract Individual Components (AIC); and AIOs that include other interaction objects, called Abstract Containers (AC).

In this context, facets are used to describe each AIC in such a way that each one of them can be composed of multiple facets. Each facet describes a particular function or behaviour that the corresponding AIC can perform in the final user interface. Concretely, there are four different types of facets: input, output, navigation and control.

Most of these recent approaches are based on the Cameleon Reference Framework [Calvary, 03] which defines the steps to be followed in order to develop user interfaces for interactive and multi-context applications. Four steps are identified:

1. Final User Interface (FUI): It represents the operational UI, the implementation in a concrete context and language, including platform details.
2. Concrete User Interface (CUI): It represents an Abstract User Interface (AUI) at an upper level for a certain context of use by means of CIOs. It is platform-independent and it is considered a reification of an upper level AUI and an abstraction of the FUI with respect to the final platform.
3. Abstract User Interface (AUI): It defines the interaction spaces (or presentation units) knitting together several tasks according to certain criteria. This is done by means of AIOs.
4. Task & Concepts (T&C): It represents the tasks to be performed and the domain concepts needed in order to carry out these tasks.

Regarding the development of collaborative systems, there are some methodologies which join interactive and collaborative aspects covering the development process of a groupware application [Garrido, 07]. AMENITIES (A MEthodology for aNalysis and desIgn of cooperaTIve systEmS) [Garrido, 05] is a methodology specially devised to study and design collaborative environments. It is focussed on the concept of group covering some important issues regarding its structure and behaviour.

Another related approach is CIAM (Collaborative Interactive Applications Methodology) [Molina, 06]. This method is based on a set of models that allow software engineers to carry out the design of user interfaces of interactive applications for collaborative teamwork.

TOUCHE (Task-Oriented and User-Centred Process Model for Developing Interfaces for Human-Computer-Human Environments) [Penichet, 07] is a complete user-centred and task-driven process model to tackle the development of UI for groupware applications starting from requirements elicitation till the implementation, taking into account the particular features and needs this kind of systems requires from the very beginning.

Comparing these different approaches, we can see that AMENITIES includes the development of the user interface in its approach explicity [Rodriguez, 07], but interaction objects are not abstracted. On the other hand, the process model defined in TOUCHE is focused on the user interface development, but it does not finally define

the toolkit elements for groupware applications. The combination of the last two proposals is the core of the work presented in this article regarding the design and implementation of user interfaces for groupware applications.

## 3 Design and Implementation of User Interfaces in Groupware Systems

The Cameleon Reference framework, and the works based on it, proposes the design of AUI according to a defined meta-model. The AUI is a reification of tasks and other concepts considered in the analysis stage. In order to devise such an abstract interface, it is necessary to identify the different AIOs, which in turn, are an abstraction of the CIOs that will be concretized in the user interface (CUI) in the implementation stage.

Our proposal is based on the conceptual model of UsiXML [Limbourg, 05b] which models the AUI taking into account facets. Facets are considered one of the last evolutions in the abstract user interface modelling for interactive systems. TOUCHE extends such conceptual model to support new features that are typical in groupware applications (see Figure 1, where new extensions are marked in grey):

- Abstract Workspace Awareness Container (AWAC) is an abstract container (AC) which supplies the shared context where the user interacts with other users and where gains awareness, i.e. it is something more complex than just a window, a frame, etc. It is an instance which shows its local changes in the rest of remote instances.
- *Embodiment*, *expressive artefact* and *visibility*. These computational techniques allow the system to show awareness, even considering present and past issues. In the design stage, they are represented as facets of the AICs (Abstract Individual Components), i.e. there will be new AICs, described by such facets, which support the awareness of the system. *Embodiment* is a facet to describe the presence of a user in a shared context. *Expressive artefact* is a facet to describe actions that are being performed by other users on the same AIC. *Visibility* is a facet to describe actions that are being performed by other users in other AWACs. As an example, a telepointer is a reification of an AIC which represents users interacting among them in an AWAC.

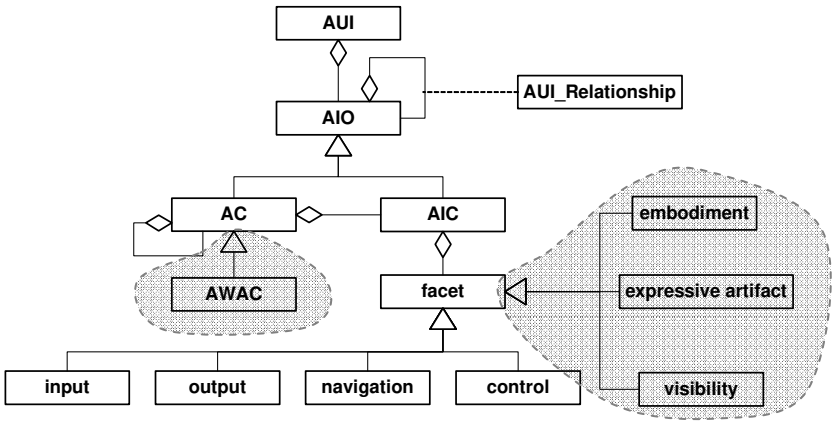


*Figure 1: Extensions to the UsiXML conceptual model for abstract UI in CSCW systems*

The implementation stage of the system tackles the generation of the final user interface (FUI) from the AIOs defined in the design stage. It is a reification process in which every component will be concretized depending on implementation details, the final platform, etc. Therefore, implementation details are considered in this stage and the CIOs, which are platform independent, are identified. Limbourg's conceptual model presents two kinds of CIOs to compose the CUI: the Graphical CIO and the Auditory CIO. Both of them could be either containers or individual components. TOUCHE extends the conceptual model proposed by Limbourg in that identification.

The CIOs in a groupware application could be single elements. They are the same that could be considered in every interactive system. However it is also possible to find composite CIOs, which could not be mixed up with containers.

Thus, CIOs to design the user interface in CSCW systems may match in the proposal of [Limbourg, 05a] considering the composite element we propose: Composite Groupware Component (CGC). That is, every single CIO may be classified as Graphical Container, Graphical Individual Component, Vocal Container, or Vocal Individual Component. A CGC has been added to the meta-model to consider a composition of such CIO.

The extended conceptual model is shown in Figure 2, where new extensions are marked in grey. Since a CGC may be composed by either graphical or auditory items, the CIO is at the same level.

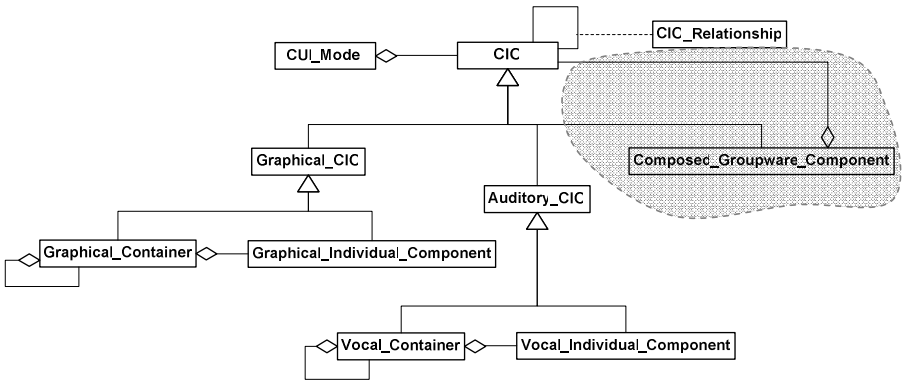


*Figure 2: Extended conceptual model for CIOs in groupware applications based on the Limbourg's one*

Examples of new CIOs for groupware applications are telepointers, avatars, and video embodiments from the embodiment facet; feedthrough buttons, feedthrough bars, and action indicators from the expressive artefact facet; and radar views, over-the-shoulder views, and cursor's eye views from the visibility facet. In [Gutwin, 98; Gutwin, 02] more information about the functionality of these items is provided.

Examples of new composite CIOs could be a set of users or a shared workspace. Figure 3 shows the first one. It is a frequent situation in applications such as chats. It is also possible to see the composition of CIOs: box, avatar, text component, and menu.

Lastly, the final user interface (FUI) includes all the necessary items for the implementation of the user interface of a groupware application. Now, these items are platform and modality dependent. The following section describes how the CIOs proposed [Penichet, 07] are implemented in a specific platform [Ibáñez, 06; Rodríguez, 07].

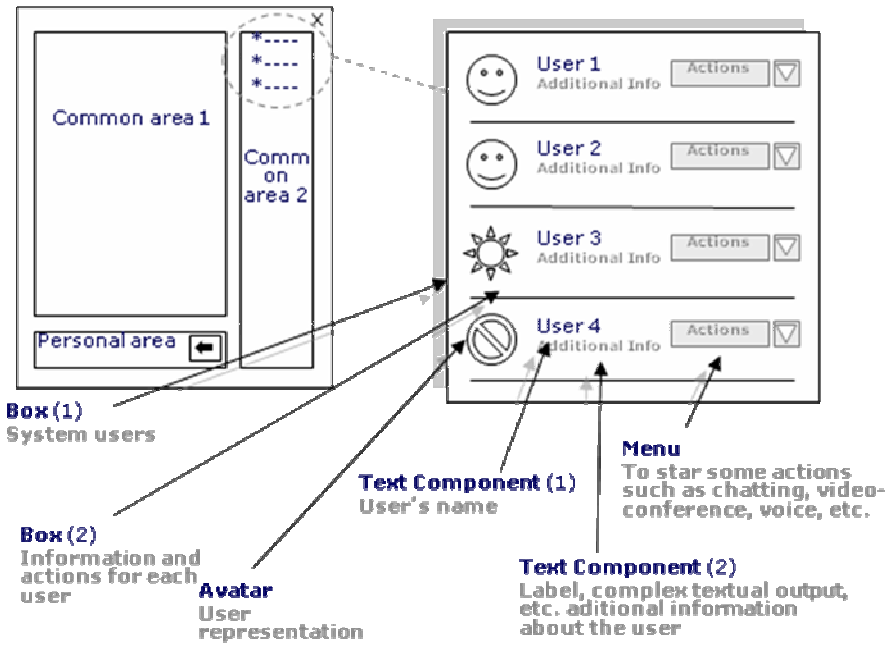


*Figure 3: Example of Composite Groupware Component: a set of users in a chat application.*

## 4 Support for the Implementation of Groupware Interfaces

A platform of graphical components (multi-user widgets) accessible through an API [Ibáñez, 06] has been designed and implemented, which simplifies the development of groupware interfaces through extensions of standard components (buttons, menus, text fields, etc.) and specific components for groupware applications (telepointers, chats, lists of online actors, etc.). The platform includes a module to dynamically manage metainformation about the system description. In particular, the metainformation module consists of the description of users, roles to be played by them, application components and user and role permissions on these components. The changes in the meta-information will produce an immediate feedback on the application to reflect the produced changes. It allows reflecting changes occurred during the execution of groupware applications.

A similar toolkit including standard widgets and groupware-specific components (but only telepointers, radar view, chat and list of participants) is MAUI (Multi-user Awareness) [Hill, 03]. All components incorporate functionalities for collecting, distributing and visualizing group awareness information. In [Gutwin, 98] it is presented a proposal focused on the study of the navigation that occurs in a shared workplace, the interaction between components, and the different views of individual and group work.

In the proposals based on the Cameleon reference framework [Calvary, 03], AIOs are identified in the design stage irrespectively of any modality of interaction (graphical interaction, vocal interaction, speech synthesis and recognition, video-based interaction, virtual, augment or mixed reality) and of the platform (software, hardware, operating system and device). The CIOs are reified from these AIOs in the implementation phase. These CIOs are still independent from the platform. In previous pieces of work like IDEAS [Lozano, 01], the CIOs are platform-dependent elements, or at least, this characteristic regarding the modality of interaction is not relevant. In this work, the effort is focused on the explicit consideration of the particular features regarding groupware applications. Therefore, the final user interface (FUI) step is joined with the concrete user interface (CUI) one.

The components proposed in the API follow the model proposed in TOUCHE, since CIOs have been developed on the basis of the AIOs characterized with the facets *embodiment*, *expressive artefact* and *visibility*. The CIOs can be replicated; their implementation assures a global consistent state in case of simultaneous interaction of several users. A replicated component will support a common state with each of their instances that are being executed in a particular moment.

A set of components has been created, as extensions of standard components, with a distributed behaviour. From these components, the CIOs shown in Table 1 have been developed with regard to the AIOs corresponding to the *expressive artefact* facet.

| **Name** | **Functionality** | **Semantic properties** | **Comments** |
|---|---|---|---|
| *DIList* | Selection, addition and subtraction of elements | Hide/read/write | Replicated/local |
| *DIComboBox* | Opening and closing of the list. Selection of one element and its corresponding highlighted | Hide/read/write | Replicated |
| *DIButton* | Pressing and releasing the button | Hide/read/write | Replicated |
| *DIToggleButton* | Pressing and releasing the button | Hide/read/write | Replicated |
| *DICheckBox* | Tick and untick of the checkbox | Hide/read/write | Replicated |
| *DITextField* | Inserting and removing text | Hide/read/write | Replicated |
| *DITree* | Opening and closing for the tree levels. Selection of one element and its corresponding highlighted. | Read/write | Replicated |

*Table 1: Description of CIOs related to AIOs with the expressive artefact facet*

It is also possible to choose between replicated or local mode when the CIO *DIList* is initialized. In the replicated mode, adding and removing elements from the

list is a distributed behaviour and therefore this fact is communicated to each instance of this component, whereas in the local mode there is no notification to any instance of the component.

Besides, the CIO depicted in Table 2 has been developed from the AIOs with the facets *embodiment* and *expressive artefact*.

| Name | Functionality | Semantic properties | Comments |
|---|---|---|---|
| *DJMenu* | Shows the user's ID who is interacting with the menu | Hide/read/write | Replicated |

*Table 2: Description of the CIO: DJMenu*

The selection of the menu items is controlled by each single user. Until a user does not close his/her local replica of the menu component another different user will not be able to open it or to interact with it. For this purpose, the menu bar shows in each moment the user who has pulled down the menu.

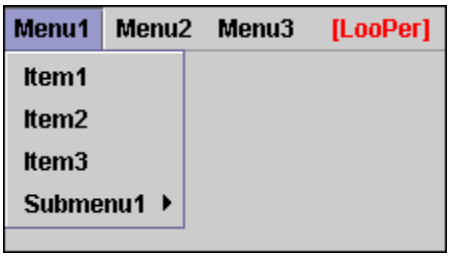


*Figure 4: Graphical representation of DJMenu*

The platform also includes specific CIOs (Table 3) for groupware applications from the AIOs characterized with the facets: *embodiment*, *expressive artefact* and *visibility*:

| Name | Functionality | Semantic properties | Comments |
|---|---|---|---|
| *DIChat* | Chat for online users | Hide/read/write | Distributed |
| *DIRoleChange* | Shows the current role and enables to change to other ones | Read/write | Replicated |
| *DICurrentRole* | Only shows active roles | Read/write | Local |
| *DIOnlineUserList* | Online users | Read/write | Replicated |
| *DIUserListRolePlayed* | Online users and the roles they play | Read/write | Replicated |
| *DIUserListSameRole* | Online users playing the same role | Read/write | Replicated |
| *Telepointer* | Pointer showing the movements of every user | Read/write | Local(write) Replicated(read) |

*Table 3: Description of the specific CIOs for groupware applications*

All these CIOs provide general information about the group structure (group awareness) of the system [Schlichter, 98].

All the CIOs developed for the platform can be classified into *simple* components and *compound* components. Compound CIOs (*DIChat*, *DIRoleChange*, *DIOnlineUser*, *DIUserLisRolePlayed* and *DIUserListSameRole*) are considered as CGCs (Compose Groupware Component). The *DICurrentRole* CIO corresponds to the Graphical Container and the *Telepointer* CIO corresponds to the Graphical Individual Component, according to Limbourg's classification.

The components within the platform provide built-in support for group awareness. In particular, this platform supports different types of awareness [Gutwin, 02; Gutwin 05]:

- Awareness of actions and intentions is the understanding of what another person is doing, either in detail or at a general level.
- Location awareness relates to where the person is working.
- Awareness of presence and identity is simply the knowledge that there are other users in the workspace and who they are.
- Awareness of artefact means knowledge about the object where a user is working on.

## 5 Case Study

We have considered a case study of a cooperative system for the decision-making of risk operations by financial institutions. In particular, a business process to grant a mortgage is addressed in which a client has applies for in a branch office [Rodríguez,

07]. This process entails the cooperative participation of people from the staff of three different organizations: a branch office, a valuation office and a notary office.

The first step in a business process to grant a mortgage consists of performing a feasibility study on the basis of a report including relevant information. In this business process the indebtedness level is calculated with the income, expenses and debts. As a result of this study, the branch office decides to pass, to reject or interrupt the operation. If the operation is initially approved, then a bank account is created.

For the sake of simplicity, the following roles will be considered: *Bank Manager* and *Head of Risk*. This fact suggests a concrete interface for each role with the tasks that can/must be performed, as depicted in Figure 5. In this example, we have taken a design decision by which tasks are represented in the user interface by means of buttons, although other components might have been also used, such as, for example, menus.

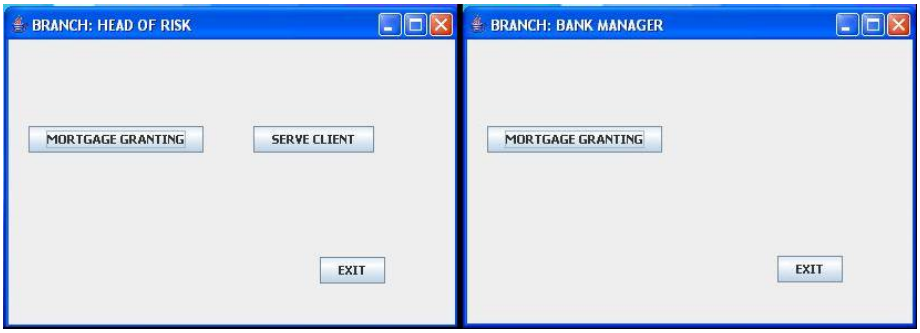


*Figure 5: User interfaces for each role*

The Branch office follows some behaviour rules. For instance, the actor playing the role *Head of Risk* can become the *Bank Manager* if the person who plays this role is absent. This change implies that the actor playing the *Head of Risk* role, when playing the *Bank Manager* role, is enabled to carry out the tasks assigned to the latter role. This requirement is translated into the user interface using the component *DIRoleChange* (1) in Figure 6. This CIO, characterized with the facets *embodiment*, *expressive artefact* and *visibility*, allows the actor to change his/her role by selecting the new role in the role panel. The component *DIRoleChange* shows, for each actor, the roles that he/she can perform in the system, the role that the actor has in each moment, and allows the user to press the *Change* button to change the role he/she is currently playing. In consequence, the user interface adapts to the new situation, showing the new *Bank Manager* role (2) and the *Give Approval* task (3), as depicted in Figure 6.

*Figure 6: Dynamically change the roles: component DIRoleChange*

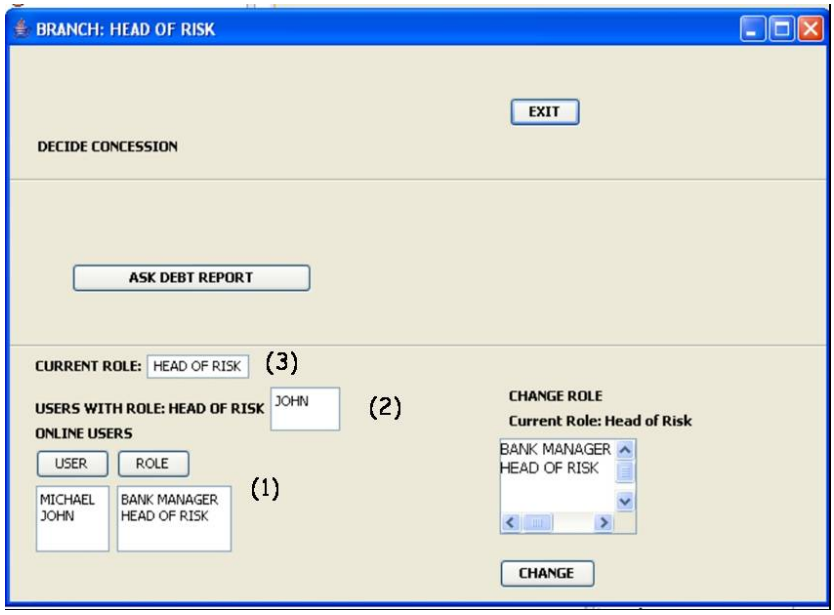


*Figure 7: Groupware components to support workspace context information*

Another example is depicted in Figure 7: the user interface of the task *Decide Concession* for the same case study. The task *Decide Concession* is performed by actors playing the *Bank Manager* and *Head of Risk* roles. This is a collaborative task because more than one participant is required to accomplish it; group awareness support is required for a more effective collaboration, e.g., how to know the actors who are using the system, where they are working and what they are doing at certain times. In order to implement this restriction, the user interface of each participating role has to contain the necessary information about the group awareness for this task. This collaboration requirement will be satisfied using two replicated components,

*DIUserListRolePlayed* (1) and *DIUserListSameRole* (2), and a local one *DICurrentRole* (3). These three CIOs are characterized with the facets *embodiment*, *expressive artefact* and *visibility*.

Awareness of presence and identity are supported by the components *DIUserListRolePlayed*, *DIUserListSameRole* and *DICurrentRole*. These CIOs provide the necessary information about the work context of the subactivity *decideConcession*.

Additionally, the collaborative task *Decide Concession* requires two specific communication requirements to accomplish it: that a shared workspace exists and that participants interact in a face-to-face manner. In Figure 8 note that the user interface of the task *Decide Concession* presents a shared workspace (the *Debt Report* (1)) and a *DIChat* component (2) to implement the interaction between the actor playing the *Bank Manager* role and the actor playing the *Head of Risk* role. The user writes a text in the local box enabled for that purpose, and presses the *Send* button, which also presents a local behaviour to each user. In the message display box with a replicated behaviour, each message is preceded by the name of the user who has written it. Finally, there is a *Telepointer* component (3) on the *Debt Report* panel showing in detail how the actor (who plays the *Bank Manager* role) carries out his activity in the system. The CIOs *DIChat* and *Telepointer* are characterized with the facets *embodiment*, *expressive artefact* and *visibility*.

The types of awareness (actions, intentions and location) are supported by the components *Telepointer* and *DIChat* and the shared workspace (the *Debt Report*). Furthermore, awareness of artefact is supported by the *Telepointer* component, which gives information regarding that the *Bank Manager* role is working on the *Debt Report* object.

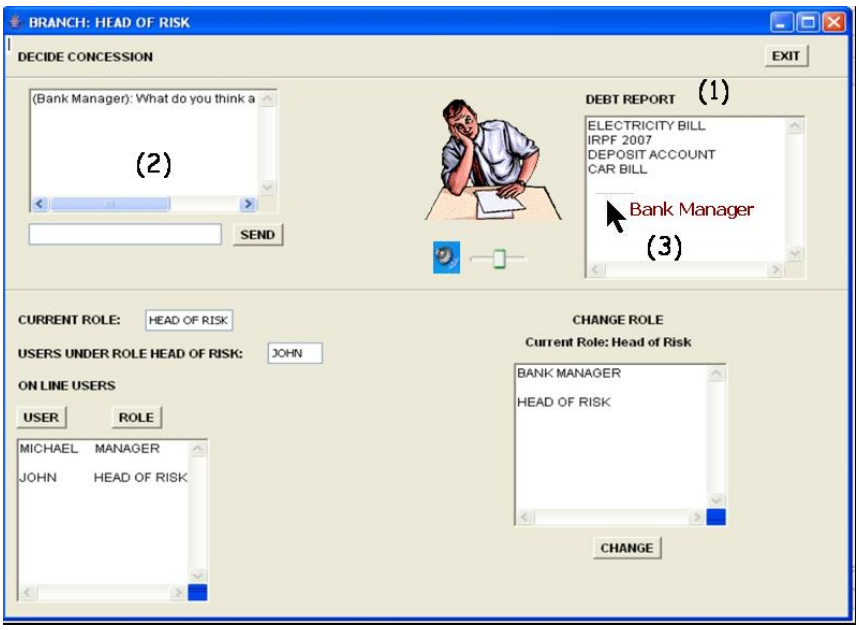


*Figure 8: User interface for the collaborative task Decide Concession*

In its current state, the proposed platform components of the present application are mainly intended to support synchronous collaboration, for example, telepointers provide continuous information about the part of the application other users are interacting with. Besides, asynchronous collaboration mechanisms are also provided. For instance, by checking the debt report panel the *Bank Manager* may see what information has been already collected to create a debt report.

## 6 Discussion

Although some approaches to groupware development may be found in the current literature (see Section 2), there is a lack of methods to tackle the development of user interfaces starting from the requirements analysis. There have been many proposals which deal with user interface development since the TRIDENT one was released [Bodart, 95], however, none of them defines a complete process model specifically devised for groupware applications. Most of them consider such development process to be started from the task analysis phase and do not take into account a previous requirements analysis stage. The vast majority of them are approaches which come from the HCI field and therefore they do not tackle with Software Engineering methodological aspects for creating computer programs. The introduced approach is based on the TOUCHE process model which starts with a requirements gathering stage and follows the typical Software Engineering methodological steps up to the implementation of the user interface.

The current development of user interfaces is based on the Reference Framework [Calvary, 03]. Present-day proposals continue extending the first TRIDENT ideas, i.e. composition, facets, etc.; however, none of them have taken into account specific aspects regarding groupware. The work presented in this paper considers groupware features in the design and implementation stages, which specifically considered, may improve the development of the final user interface of applications to be used by group of users in order to perform common tasks.

Some implementation methods in groupware applications [Gutwin, 02; etc.] are closer to the final implementation of user interface widgets rather than to methodological approaches within a whole process model. The piece of work we are presenting also tackles the implementation of the user interface, but it is integrated in the TOUCHE process model, that covers from requirements gathering till the user interface implementation.

## 7 Conclusions

The development of groupware applications in which the user as a member of a group, is the fundamental piece of a system, is gaining relevance. Users interact with the application, but they also interact among them through the application itself. Groupware applications are becoming more and more widely used thanks to different factors: the improvement of the network infrastructure, communications, and development tools, among others. Most of the current applications have elements regarding collaboration, coordination, or communication among its different users;

thereby, the user needs to be aware of the presence of the other users in the system: what they do, what they did, how they collaborate, etc.

This work presents a proposal for the development of user interfaces of groupware applications which is based on existing methodological frameworks. Accordingly, the work starts from two previous proposals (i.e. AMENITIES and TOUCHE) which complement one another in order to accomplish the design and the implementation of this kind of user interfaces. This proposal encompasses the UI design based on abstract interaction objects, which provide modality and platform independency, and its subsequent implementation using a specific platform.

We have presented a framework which allows developers to implement featured, abstract UI for groupware applications by means of a toolkit that supports different types of awareness (location, action, intentions, presence, identity and artefact); it allows anticipating actions and reduces the need to coordinate tasks and resources. Nowadays the platform is being applied to the development of collaborative games based on association of concepts for children.

Currently, the future work includes the precise definition of the process to be followed with the presented proposal. In particular, a deeper analysis of the facets in abstract models would result in a better selection of UI components on the basis of their finer-grained classification.

**Acknowledgements**

We would like to acknowledge the national projects with reference CICYT TIN2008-06596-C02-01 and CICYT TIN2008-05995/TSI, the Junta de Comunidades de Castilla-La Mancha PAI06-0093-8836 regional project and the Comunidad Autónoma de Castilla-La Mancha, Consejería de Educación y Ciencia PAC07-0020-5702 project, for partially funding this work.

## References

[Bodart, 94] Bodart, F., Vanderdonckt, J.: On the problem of selecting interaction objects. In Proceedings of the conference on People and computers IX. Cambridge University Press New York, NY, USA. ISBN: 0-521-48557-6. Pp. 163 – 178. Glasgow, 1994.

[Bodart, 95] Bodart, F. et al: Towards a Systematic Building of Software Architectures: the TRIDENT Methodological Guide. In Proc. Of Eurographics Workshop on Design, Specification, Verification of Interactive Systems. DSV-IS'95. Eurographics Series, pp. 237-253. June 1995.

[Calvary, 03] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers. Vol. 15, No. 3, pp. 289-308. June 2003.

[Dix, 98] Dix, A., Finlay, J., Abowd, G., Beale, R.: Human-Computer Interaction. Prentice-Hall 1998.

[Dourish, 92] Dourish, P., Belloti, V.: Awareness and Coordination in Shared Workspaces. Proceeding of the ACM Conference on Computer-Supported Cooperative Work, pp. 107-114. 1992.

[Garrido, 05] Garrido, J.L., Gea, M., Rodríguez, M.L.: Requirements Enginnering in Cooperative Systems. In Requirements Engineering for Sociotechnical Systems, Idea Group, Inc., pp. 226-244. USA 2005.

[Garrido, 07] Garrido, J.L., Noguera, M., González, M., Hurtado, M.V., Rodríguez, M.L.: "Definition and use of Computation Independent Models in an MDA-based Groupware Development Process", Science of Computer Programming 66, 2007, pp. 25-43.

[Gutwin, 98] Gutwin, C., Greenberg, S.: Design for Individuals, Design for Groups: Tradeoff between power and workspace awareness. Proceeding of ACM Computer Supported Cooperative Work 1998. pp. 207-216. 1998.

[Gutwin, 02] Gutwin, C., Greenberg, S.: A Descriptive Framework of Workspace Awareness for Real-Time Groupware. Journal Computer Supported Cooperative Work. Volume 11. pp. 411-446. 2002.

[Gutwin, 04] Gutwin, C., Penner, R., Schneider, K.: Group Awareness in Distributed Software Development. Proceeding of ACM Computer Supported Cooperative Work 2004. pp. 72-81. 2004.

[Gutwin, 05] Gutwin, C., et al.: Supporting Group Awareness in Distributed Software Development. In: Bastide, R., Palanque, P., Roth, J. (eds.) Engineering Human Computer Interaction and Interactive System. LNCS, vol. 3425, pp. 383-397. Springer, Heidelberg 2005.

[Hill, 03] Hill, J.M.: A Direct Manipulation Toolkit for Awareness Support in Groupware. Thesis, University of Saskatchewan. 2003.

[Ibáñez 06] Ibáñez Santórum, J.A.: Design and Implementation of a Platform for the Development of Groupware Systems. Bachelor's degree thesis. Department of Languages and Computer Systems. University of Granada, 2006. (In Spanish)

[Limbourg, 05a] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López Jaquero, V. UsiXML: a Language Supporting Multi-Path Development of User Interfaces, Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004 (Hamburg, July 11-13, 2004). LNCS, Vol. 3425, Springer-Verlag, Berlin, Germany, 2005.

[Limbourg, 05b] Limbourg, Quentin: Multi-Path Development of User Interfaces. PhD Thesis. Université catholique de Louvain. Belgium, 2005.

[Lozano, 01] Lozano, M.D.: Object-Oriented Methodological Environment for the Specification and Development of User Interfaces. PhD Thesis. Universidad Politécnica de Valencia, 2001 (In Spanish).

[Molina, 06] Molina, A.I., Redondo, M.A., Ortega, M.: Conceptual and Methodological Framework for Modelling Interactive Groupware Applications. Groupware: Design, Implementation and use. pp. 413-420. LNCS 4154, Springer-Verlag 2006.

[Penichet, 07] Penichet, Victor M. R.: Task-Oriented and User-Centred Process Model for Developing Interfaces for Human-Computer-Human Environments. PhD. University of Castilla-La Mancha. 2007.

[Rodríguez, 07] Rodríguez, María L., Garrido, J.L., Hurtado, María V., Noguera, M.: An Approach to the Model-based Design of Groupware Multi-User Interfaces. Groupware: Design, Implementation and Use. Lectures Notes in Computer Science. Volume 4715, pp. 157-164, 2007.

[Schlichter, 98] Schlichter, J., Koch, M., Bürger, M.: Workspace Awareness for Distributed Teams. In: W. Conen, G. Neumann (eds.), Coordination Technology for Collaborative Apllications, Springer Verlag, Heidelberg, 1998.

[da Silva, 00] da Silva, P., Paton, N.W.: UMLi: The Unified Modeling Language for Interactive Applications, in Conf. Proc. Of UML00, UK, pp. 117-132, 2000.