Validation of a Framework for Enriching Human–Computer–Human Interaction with Awareness in a Seamless Way

Montserrat Sendín1∗, Juan-Miguel López-Gil2 and Víctor López-Jaquero3

1 GRIHO HCI Research Lab., University of Lleida, E-25001 Lleida, Spain
2 GALAN Research Group, University of the Basque Country UPV/EHU, 01006 Vitoria-Gasteiz, Spain
3 LoUISE Research Group, University of Castilla-La Mancha, 02071 Albacete, Spain
∗Corresponding author: msendin@diei.udl.cat

Computer supported cooperative work (CSCW) environments enable users to interact with each other by using computers to conveniently share relevant data across the user interface. Awareness is an essential requirement in CSCW to convey precise information about the context in which the work in group is taking place, contributing thus to collaboration between users. In this kind of environments, we need to go beyond traditional human–computer interaction to embrace human–computer–human interaction (HCHI). It is necessary to devise flexible mechanisms to support HCHI in dealing with the diversity of contexts and group concerns. Furthermore, these mechanisms should endeavor to provide a seamless integration with current development techniques. This work presents a multi-purpose framework to include group awareness in HCHI systems in a seamless way. Proposed framework is based on the Dichotomic View of Plasticity approach. An experiment was conducted with two different versions of a specific groupware platform: the original platform and a new version of it, extended by means of the proposed framework. The goal was twofold: (i) to verify the benefits of applying this framework and (ii) to validate, in terms of user satisfaction, the improvement regarding groupware features introduced in the extended version. The results of the experiment backed up our hypothesis by showing that proposed framework is able to add awareness support to existing human–computer–Human (HCH) interfaces in a seamless way. It is also showed that added awareness components effectively contributed to achieve a higher level of collaboration among users.

RESEARCH HIGHLIGHTS

• A software infrastructure to deliver and support enriched applications with awareness.
• An experiment to validate the infrastructure in a groupware application.
• An analysis of how the awareness features introduced have an impact on users’ perception.

Keywords: human computer interaction; computer supported cooperative work; empirical studies in collaborative and social computing

Editorial Board Member: Françoise Détienne

Received 30 November 2012; Revised 15 May 2013; Accepted 31 July 2013

1. INTRODUCTION

Human–computer–human interaction (Fuller, 1997) (HCHI) is committed to increase the interaction among users by appropriately mediating or managing group activity. HCHI is a specialization of human–computer interaction for computer supported cooperative work (CSCW). Cooperative work has been identified as a phenomenon we can study systematically, as a category of work practice. Moreover, in CSCW is also paramount the coordinative practices, including their methods and techniques. The most challenging problem for CSCW is ‘developing technologies that enable ordinary actors to construct computational devices so they can (a) regulate their own complex cooperative activities more efficiently, safely, dependably, etc. and (b) manage the regulation of their activities’ (Schmidt, 2011). A systematic analysis of the complexities of
CSCW concept is available at Schmidt (1998). To be more precise, HCHI is focused on understanding characteristics of interdependent group work with the objective of designing adequate User Interfaces (UIs) to support work-in-group in CSCW systems. It is widely known that HCHI is only really successful when the UI incorporates specialized interactive widgets that help encouraging work in group, that is, making sure that during the interaction with this kind of systems, users never lose the overview of the working group. User tasks should be carried out in a given shared context (Alarcón et al., 2005), similarly to the way that they are performed in the real world.

According to that, if we are interested in making the context for the whole shared activity easily assimilated by all the users, the introduction of awareness (Gutwin, 1997) and work in group support becomes an essential requirement. In this sense, it is necessary to devise mechanisms to support HCHI in a way flexible enough to cope with the diversity of contexts of use related not just to group concerns but also to assorted computing platforms and environments in which group members can be working. Moreover, these mechanisms should be able to be integrated in a seamless way, independently of the underlying collaborative platform or development technique.

One of the problems that a designer faces when dealing with awareness is the many and complementary definitions found in the literature. Some problems are also derived from the fact that this concept has been used to designate when one person is aware of something, and when a person is aware of something even not noticing it (Schmidt, 2011). Although many different types of awareness are considered in the literature (Gross et al., 2005), this work is focused on the shared-knowledge awareness (SKA), a special kind of awareness representing the understanding that the members of a group have about their shared knowledge, that is to say, all of the aspects regarding their collaborative work (Collazos et al., 2002). We are referring to aspects such as the organizational context, pieces of information of each group member, their contributions to common tasks and any group-related constraints and shared knowledge. Apart from SKA, there are other types of awareness that can also be used in our framework to describe awareness provision, as they are not necessarily disjoint sets. For example, Activity Awareness (Carroll et al., 2006) or Workspace Awareness (Gutwin and Greenberg, 2002) could be considered. Actually, they are also indirectly taken into account and they can be particularized during the design of awareness provision. Although SKA was first introduced in the context of collaborative learning (CSCL), in recent years there has been a steady shift in CSCW and CSCL systems towards a complementary middle ground, resulting in the fact that both fields tend to converge. Consequently, mechanisms till now accepted only for one of these two fields have also started to be applied for the other field (Leetaru, 2009).

We are interested in this type of awareness because it provides a context for group activity, which, conveniently assimilated by all the group members, allows tackling activities effectively and efficiently in a collaborative manner. The final goal for using the presented framework is precisely helping users to make common tasks collaboratively. However, this kind of awareness requires sophisticated data capturing and processing capabilities. The integration of all these aspects in the development of HCHI is not a trivial task. An ideal infrastructure should provide the capacity to adapt interactive widgets and data to each particular context with little or no extra effort from the UI designer. In this paper, we present how the approach and software infrastructure described in the Dichotomic View of Plasticity (Sendin and López, 2009) can be applied in HCHI to build a context for the group activity, thus offering support for the different aspects of shared-knowledge awareness. An experiment was conducted to validate the benefits of applying this framework to improve the facilities for work in group in human–computer–human (HCH) interfaces for CSCW systems. The results of the experiment showed that the proposed framework is capable of adding new awareness capabilities to existing human–computer–human interfaces that do not include an appropriate work in group support. By analyzing the results collected during the experimentation, we concluded that these new groupware features contribute to a better collaboration among users. Furthermore, the experiment also shows that the extra awareness support is integrated seamlessly into the original application.

This paper is structured as follows. The next section summarizes the related work. Section 3 describes the aforementioned Dichotomic View of Plasticity approach. Design and implementation bases of the software infrastructure to support HCHI are introduced next. Then, the experiment conducted is fully described. Finally, some conclusions and future work end the paper.

2. RELATED WORK

HCHI development has been an active research field for a long time. It has been long recognized that successful work in group is not simply based on the union of individual tasks, but an organized set of coherent activities with good strategies of communication, cooperation and coordination among group members (Alarcón et al., 2005). Awareness describes the necessity for obtaining some level of shared knowledge about each other’s activities that individuals working together have (Dourish and Belloti, 1992). It reduces the meta-communicative effort required to collaborate across physical distances in distributed environments and promotes real collaboration (Palfreyman and Rodden, 1996). It is also critical to improve the collaboration efficiency of a group, especially when teammates are geographically separated. Simple awareness features, such as the last web pages visited or past web activities, have been proven useful for group collaboration (Wang et al., 2012). On the other hand, group awareness information, supported through a graphical language and a window showing the participants’ presence, has proven be successful in collaborative learning situations (Romero-Salcedo et al., 2004). Other well-known
examples of awareness mechanisms are the use of telepointers, participant lists or multi-scroll bars (Gutwin et al., 1996).

Despite of these advances, working together through a groupware system often seems inefficient and clumsy compared with face-to-face work (Gutwin and Greenberg, 2002). In nearly all collaborative work settings, an ‘infrastructure’ is in place to support the articulation of individual activities in significant ways, such as providing awareness and facilitating communication (Xiao and Seagull, 2007).

Many authors have identified and pointed out the relevance of including different elements and new mechanisms aimed to improve working group collaboration by providing support to communication, coordination and cooperation processes (Yankee Group, 1995). However, from an organizational point of view, there are other working group particularities that should be also considered. These include the definition of organizational structures, the collaborative workflow management or questions regarding the adaptation of the collaborative system to the heterogeneity of needs and particular preferences from each working group member (Ackerman, 2000). Despite some works, such as the AMENITIES system (Garrido et al., 2007) or the framework presented in Lópe Antonaya and Bravo Santos (2010), tackle these kinds of questions, most proposals do not offer an integrating perspective, or they do it without the required analysis and further study.

Systematic or computational support for modeling awareness is hardly provided in the literature (Sendín and López, 2011). Support for awareness is still mainly addressed in an ad-hoc manner, and even more important, in most cases developers must rebuild it from scratch for every new system (Palfreyman and Rodden, 1996; Sendín and López, 2011). Thus, it implies significant costs and inconvenience (Iqbal et al., 2003). Nevertheless, we can name some works relevant for having made an effort in this line, such as Figueroa-Martinez et al. (2012); Penichet et al. (2010) and (Gallego et al. (2011)). In particular, the work by Simone and Bandini (2002) can be considered as an exception, as it provides a systematic support for modeling awareness. It proposes a reaction-diffusion metaphor to describe the awareness phenomenology and to take into account the integration of two specific types of awareness: (i) by-product awareness—generated in the course of the activities people must do in order to accomplish their cooperative tasks—and (ii) add-on awareness—the outcome of an additional activity. The model of awareness derived from the metaphor makes visible and accessible by different types of users a set of elemental primitives whose flexible composition allows them to dynamically construct the awareness mechanisms they need. One of the practical applications of this work has been the conception of a computational system that can promote awareness to achieve a better coordination and committed pathway inclusion in daily clinical practice (Cabitza et al., 2007).

It is worth mentioning that due to the variety of needs that can arise from participants as well as the fact that group decisions should be taken under a shared-knowledge perspective, there is a major necessity for adaptability in the HCHI scope, even more than in the case of traditional single user software. However, most cited systems are only focused on static user aspects, providing different kinds of prefixed profiles. Part of the works realized in this line have proposed new models of adaptable systems based on the component-based development paradigm and the composition processes. Thus, the CocoWare.Net (Slagter, 2001) platform provides a set of components, mostly destined to provide different communication mechanisms that users can combine to modify the system. Another interesting approach is the system EVOLVE (Stiemerling and Cremers, 2000), which provides a model based on the connection by means of ports with which it is possible to construct adaptable UIs.

Apart from user needs, it is also necessary to consider device heterogeneity. In general, groupware frameworks that have been designed explicitly to support mobile or heterogeneous devices usually focus on specific application domains. They include meeting environments, where mobile devices are used as input devices to share data. A study (Guicking and Grasse 2006) references some interesting systems, such as Pebbles (Myers, 2001) and SharedNotes (Greenberg et al., 1999), and also collaborative face-to-face learning environments, including ConcertStudeo (Wessner et al., 2001). Although these proposals have shown to be useful to support each one of the specific collaborative activities, they are not valid as general solutions.

Taking the implementation aspect into consideration, a wide range of tools and solutions to develop collaborative applications has arisen. As a result, there are a lot of different approaches, perspectives and criteria existing today. For example, there is a wide variety of frameworks that provide comprehensive support for the development of groupware applications. We can mention the works performed by Marsic: DISCIPLE (Marsic, 1999) and Manifold framework (Marsic, 2001). Another interesting work is Agilo (Guicking and Grasse, 2006). However, a weakness can be pointed out: it needs to improve dynamic characteristics support in an application-independent way, such as context-awareness and mobility-like factors. Finally, considering the distribution strategy regarding the underlying networking topology (basically Client–Server and Peer-To-Peer distribution models), we can conclude that most approaches are highly dependent on the server and therefore liable to the wireless connectivity instability, implying that some problems like network failures would entail communication problems. Furthermore, they do not support the autonomy and interoperability capabilities required by a mobile environment. The opposite alternative related to the distribution criterion is a fully distributed approach under a totally decentralized strategy. The fully distributed approach has certain drawbacks and limitations, such as maintenance of shared information coherence, which needs to be highly replicated, generating a lot of overhead. Furthermore, there is
no global group perspective, and therefore there is no possibility of an overall inference useful for the group.

It is clear that a certain degree of decentralization is required. Maybe the best solution is to balance both alternatives, trying to obtain a fair decentralization. Our approach, based on the Dichotomic View of plasticity, follows the client–server distribution model. However, instead of adopting a fully server-centered approach, it looks for a balanced distribution of responsibilities between both sides in order to obtain the level of autonomy pursued. This approach, as well as the software infrastructure proposed to support shared-knowledge awareness in HCHI, is presented in the next section. As a relevant point, it is worth mentioning that the software infrastructure has been designed with the aim of introducing awareness components in a seamless way.

3. APPROACH FOR ENRICHING HUMAN–COMPUTER–HUMAN INTERFACES

Our approach is grounded on the concept of plasticity. Plasticity is defined as the capacity of a UI to overcome variations in the context of use without degrading usability, in an economic and ergonomic way (Calvary et al., 2002). Plasticity must not be confused with the malleability concept. Plasticity is a concept belonging to the human–computer–interaction field and directly related to usability. It refers to the degree of adaptation of the UI to different circumstances or contexts of use. Malleability is a concept influenced by the software engineering and the CSCW fields. It refers to the capacity of a system to be customized to different user requirements, making it capable of solving different usages or, more specifically, different group tasks or a variety of collaborative facilities (e.g. call centers, mailboxes, shared repositories and role-based collaboration) (Lee and Prakash, 1998).

One of the approaches available to achieve plasticity is the Dichotomic View of Plasticity (Sendín and López, 2009). It offers useful guidelines and software support to seamlessly embed an appropriate support for automatic adaptations of the UI upon changes in the context of use. One of its applications is enriching HCHI by embedding awareness support in groupware systems (Sendín and López, 2011). In this case, the adaptation is focused on adapting interactive widgets and data according to the overview of the working group. To do so, it integrates and exploits group awareness issues as an integral part of the adaptation process. This approach is based on client–server architecture, which provides a communication mechanism between target human–computer–human (HCH) interfaces and the server along the entire collaborative task.

According to the Dichotomic View of Plasticity (DVP) (Sendín and López, 2009), there are two types of adaptations: implicit and explicit. When the complexity involved in the adaptation process is too high to be handled on the target client platform, the client asks the server for an explicit plastic adaptation, which implies a higher level of reasoning (e.g. adaptations required when the user switches from using a smartphone to a tablet during the same session, or in the case of groupware environments when it is necessary to infer global group properties from a shared knowledge). On the one hand, if the complexity of the adaptation or awareness mechanism to be applied is low enough to be solved locally, the client tackles it by providing an implicit plastic adaptation (e.g. a screen brightness adaptation triggered when using a mobile device and the light conditions change). In other words, explicit plastic adaptation is generally associated to major changes in the UI, caused by unforeseen or simply more complex situations that involve either a reconfiguration of the UI or a global inference process (this last is the case for groupware environments).

In any case, due to the considerable scope involved, it needs to be solved in a server, where it is brought into operation under an explicit request by the client. This is why we call it ‘explicit’. For instance, in groupware environments, such as it is going to be detailed henceforth, an explicit request is produced when a group member needs to share some group circumstance detected, which can affect the overall group. It has to be taken into account that is in the server where is maintained a sort of group memory. Furthermore, it is in the server where there is the necessary infrastructure to manage an inference process.

On the other hand, implicit plasticity is in charge of providing proactive adaptations (also called adaptivity) at runtime, as the user goes through new contexts of use. It tackles specific modifications in the UI, originated by predictable contextual changes, which can be solved by an automatic readjustment on the client side, without any express action or request. This is why we call it ‘implicit’. For instance, in groupware environments, it is necessary becoming aware of each interaction with the corresponding teammates. Implicit plastic adaptation is carried out locally by means of a specific software component, the so-called Implicit Plasticity Engine (IPE), previously embedded on the target client UI. Resulting UIs augment their adaptation capabilities and work in group support, thus helping to do tasks in a collaborative manner.

Figure 1 shows an overview of the process involved in an explicit plastic adaptation, where the contextual information has to be sent to the server. The components that characterize the context in the Dichotomic View of Plasticity are as follows: the user, the platform where the application is being executed, the physical environment where the interaction is taking place, the current task and the so-called particular group-awareness—(Pga)—depicted from left to right at the top right in Figure 1. Pga gathers the perception that each group member has about the activities performed by the rest of the group, as well as the particular understanding of group circumstances that each member has. It is gathered locally in client platforms and intervenes in the processing of local adaptations and awareness mechanisms.

In order to achieve a real collaboration, it is essential that each member shares with the group his/her individual
All of these functions are managed by the Awareness Manager, the network usage identified by Correa and Marsic (2003); (ii) achieving a trade-off between the degree of awareness and operational balance between both sides, in the sense of individual perception about the group. Group-awareness is required because of relevant changes in its constraints and implications during the server inference process. All of these functions are managed by the Awareness Manager component in the server—see Figure 1. The results of this process can consist in specific adaptations or in inferring global properties. Finally, the server conveniently delivers these results to the target HCH interface customized for each group member by means of a set of adaptations or properties particularly adjusted to the overall group situation. By this last step, global considerations beneficial for the group are transmitted back to the group members contributing on a high level of collaboration. In this approach, the client—in particular the group members contributing on a high level of collaboration—resorts to the server when sharing his/her particular group-awareness is required because of relevant changes in its individual perception about the group.

This infrastructure provides a series of benefits: (i) an operational balance between both sides, in the sense of achieving a trade-off between the degree of awareness and the network usage identified by Correa and Marsic (2003); (ii) autonomy to provide adaptation and awareness mechanisms on the client side; (iii) a real-time reaction to contextual variations, contributing both to a proactive adaptation and to a fluent communication and coordination between group members (Sendín and Collazos, 2006) and (iv) two levels of awareness: under a local and a global perspective, which conveniently combined enrich the shared-knowledge awareness. Moreover, there are other important benefits regarding design issues since, as explained along the paper, capabilities related to awareness are embedded into existing HCH interfaces in a seamless way regarding their core functionalities.

Next section describes the software infrastructure to be embedded into those HCH interfaces characterized by poor or no awareness support, with the aim of enriching their collaboration capabilities. It also describes the operation of the system on the server side.

4. SOFTWARE INFRASTRUCTURE TO DELIVER AND SUPPORT HCHI ENRICHED WITH AWARENESS

This section presents the software infrastructure applied in the client side in order to generate the IPE (Implicit Plasticity Engine), a specific component for a particular client application. Then, the server side, composed of the Awareness Manager and the Delivery System components, is described.

4.1. IPE Software Structure: General Guidelines for Collaborative Scenarios

The Implicit Plasticity Engine is a runtime adaptive engine to provide proactive adaptations and awareness mechanisms to the target HCH interfaces in the client side. In order to illustrate the different effects of this architecture in a particular case, a specific UI for a groupware system has been used as a running example throughout the rest of this section. The chosen example is called Working Controller, a collaborative application used in a construction company, whose main objective consists in supporting workers during their working day by controlling the status of their current work and the work of their co-workers.

The software architecture of the IPE has been designed based on design principles such as transparency and reusability. Thus, we aimed at supporting multi-purpose usage. One of the main points to reach flexibility and reusability is that adaptive mechanisms and system core functionality are handled orthogonally, allowing them to evolve individually. Managing orthogonality needs to apply some kind of separation of concerns technology. The importance of ‘separating concerns’ when dealing with adaptability is pointed out in Han et al. (1998), where workflow adaptations are classified according to four levels of abstractions: Infrastructure, Resource, Process and Domain, as it is also pointed out in Divitini and Simone (2000). Based on these premises, the IPE follows a software architecture consisting of three layers, as depicted in Figure 2.

The logical layer comprises the application core functionality, that is, the original application, which is supposed to be provided with poor work in group support. In our example, that would be the original code for the Working Controller application. The context-aware layer includes a representation of the group constraints. It models all the information generated during the execution of the collaborative tasks that could affect the
whole group state. For example, in the *Working Controller*, it includes information regarding when a task is finished or when a worker runs out of a material. Additionally, it logs interactions of the user with the other group members, as well as any data related to communication and coordination actions (e.g., the e-mails sent). All this locally gathered information conforms an individual understanding of the group, which is represented by the *particular group-awareness* component. This information is kept updated for further use, acting at two levels of awareness: (i) in local decisions by means of awareness mechanisms (e.g. if a required material A is run out, a notification to the management department must be sent) and (ii) since it is shared with the rest of the group to shape the *shared-knowledge awareness* (SKA), it makes also global decisions that can affect the evolution of the working group. In our example, the fact that a certain material needs to be resupplied is gathered in the SKA. Consequently, this information can result in global group decisions (e.g. if it is observed that the material A runs out too often, it can be decided that it needs to be resupplied in a larger amount).

Finally, the intermediate layer is responsible for applying the adaptations and awareness mechanisms to the core system (*logical layer*), according to the *particular group-awareness* component (*context-aware layer*). In the *Working Controller*, this layer acts, for example, by automatically notifying to the management department when a certain material is run out. In other words, this layer acts as a transparent link between the other two layers.

According to the approach and guidelines defined in Sendín and Collazos (2006), we use Aspect-Oriented Programming (AOP) as the separation of concerns technology chosen for reaching orthogonality and transparency applying awareness mechanisms. It is worth mentioning that this programming paradigm has been successfully used in other environments where Dichotomic View of plasticity has been applied (López and Sendín, 2010; Sendín and López, 2009). This is why the intermediary layer is named (3) *aspectual layer*. Thereby, it is composed of program units called *aspects* that are in charge of intercepting the operative of the core system to

---

**Figure 2.** Generic IPE’s software structure for collaborative systems.
apply the augmented functionality related to group issues in accordance with the current state of the particular group-awareness. To be more precise, it is responsible for (i) capturing the context to conform each group member’s particular perception (this is obtained by monitoring those operations involving other group members—e.g. cancelling a task) and (ii) triggering some awareness mechanisms aimed at promoting coordination and communication between users according to the particular group-awareness. These kinds of actions, destined to promote collaboration, are strategically triggered by means of AOP mechanisms and then are newly reflected in the particular group-awareness, thus providing feedback to the local information about the group. The architecture of the aspectual layer follows a pre-established arrangement of components inspired on the groupware principles by Ellis et al. (1991), regarding the three fundamental aspects involved in the interaction between group members. These three aspects are Communication, Coordination and Collaboration, known as the three ‘C’s of groupware.

When implementing these issues in our architecture, taking into account that each one of these goals is usually materialized in different considerations and then in different actions to be done, a specific program unit (an aspect unit) with a treatment differentiated is used. It is intended to contribute to each purpose in a specialized way, following the orthogonality and reusability guidelines. Thus, the aspectual layer consists of the Communication aspect, the Coordination aspect and the Collaboration aspect.

Moreover, according to Dourish and Belloti (1992), the SKA must be collected, gathered and distributed to the rest of the group members in a passive manner, rather than explicitly provided by the actors. This is what the Collaboration aspect does by sending pieces of information to the server in an implicit manner. This action triggers an explicit plastic adaptation, thus delegating the decision about the adaptation to the server.

Figure 2 depicts a simplified sketch of the software structure for a generic IPE for collaborative systems. Apart from the already presented components in the aspectual layer, the core Application represents the business code of the underlying system. The particular group-awareness component in the context-aware layer models the user understanding about the group.

More details about this component can be found in Sendín and Collazos (2006) and Sendín and López (2011). In the next section, the infrastructure for the server component is described.

4.2. The Server Component: Awareness Provider

The Awareness Provider is the server component that supports and delivers HCH interfaces enriched with awareness elements. It is composed of two components: (i) the delivery system, which is in charge of preparing and delivering the HCH interfaces generated and (ii) the awareness manager, responsible for (i) constructing and managing the SKA and (ii) inferring not only adaptations, but also the convenience of triggering awareness mechanisms at runtime.

4.2.1. The Delivery System

Figure 3a depicts how the delivery system works. Once the HCH interface and the whole application—the Working Controller in our example—has been uploaded into the system—step 1 in Figure 3a, the IPE derivation tool processes it—step 2 in Figure 3a. This process consists in, first of all, instantiating the implicit plasticity framework (IPF)—presented hereafter—for the particular context of use and the requirements of the target HCH interface. A specific component for supporting awareness is obtained, consisting of both the context-aware and aspectual layers—step 3 in Figure 3a. Then, this component is woven together with the original code from the target application—step 4 in Figure 3a, obtaining the expected IPE—step 5 in Figure 3a. Thus, specific awareness support is embedded into the original application. All these components make up the builder of awareness components subsystem.

Finally, as the last step in the process, generated IPE is registered in the server and made available in the repository of groupware applications, so that the client is able to download and install it in the target platform—step 6 in Figure 3a. It is worth mentioning that the instantiation process for the IPE derivation is performed in a semi-automatic way, that is, in collaboration with the IPE designer. More details regarding this process are available in Sendín and López (2011).

The IPF component is a generic and white-box client-side application framework ready to be instantiated into specific components to support awareness features for particular groupware applications (the so-called IPE components). Therefore, the IPF is used by the builder component to prepare interfaces enriched with awareness support. Because of its level of genericity and flexibility in the code structure, the effort required by the designer to customize whatever particularity found (such as different application domains, awareness mechanisms, group features, context of use needs, etc.) is reduced.

It is worth mentioning that this level of flexibility has been reached by encapsulating specific hooks and dependences with the logical layer in well-identified pieces of code, which are declared abstract in the framework. We are referring to both abstract methods and abstract pointcuts. Furthermore, and what is the most important; thanks to the intrinsic AOP mechanisms, components instantiation is made without recoding the original code and software structure (the logical layer) at all. In short, it operates in a totally seamless way with the target application.

4.2.2. The Awareness Manager

Figure 3b displays how the Awareness Manager component works. It gathers all the group work parameters received from the individual perceptions of the participants in the collaborative...
task, which are modeled by the so-called SKA. Thereby, this component is in charge of maintaining the consistency of the distributed shared knowledge. By applying artificial intelligence techniques, it supports decision-making by means of an inference engine specially designed, which manages the different group constraints.

Thus, when a particular client sends a request to the server (i.e. when it is necessary to share some group circumstances, such as when a collaborative task is cancelled), the information regarding awareness is delivered to the inference engine—step 2 in Figure 3b—in order to determine which adaptation is the most suitable one to be applied to the HCH interface, given the current group situation—step 3 in Figure 3b. During this process, some global properties and overall considerations in the benefit of the group can be deduced (e.g. making suggestions about completing cancelled tasks). The results are then sent back to the clients expressed as a set of adaptations in conformance to the inferred global properties—step 4 in Figure 3b. More details regarding this mechanism are available in Sendín and López (2011).

5. EMPIRICAL VALIDATION OF AWARENESS SUPPORT IN UIS

In this section, an experiment conducted to validate the approach and the software infrastructure introduced in previous sections is described. Once the experiment context and the procedure followed to conduct it are described, the results collected are analyzed.

5.1. Experimental Context

It is widely accepted that including awareness features affects the execution of collaborative tasks (Ellis et al., 1991). By conducting this experiment, we aim at validating that awareness mechanisms resulting from applying the software infrastructure presented in this paper are valid to support HCHI enriched with awareness and that they contribute to achieve a real collaboration. Another relevant conclusion drawn from the experimentation is that by embedding the software infrastructure presented in this paper into the target groupware...
application, the original application gets enriched with new awareness features without recoding the original code at all.

For this experiment, two versions of the same groupware tool, namely Lucane (Fiack, 2013), were used by the participants. One of the versions was left unchanged, while the other one was enriched to include some extra awareness support features by using our framework. The participants were asked to perform the same collaborative tasks in both versions of the tool. Latter, they were asked to fill in some questionnaires in order to find out the effects that including the extra awareness features had in the work in group. Therefore, our null hypothesis (H0) for the experiment was that both versions of the tool would yield similar results after analyzing the questionnaires answers. H1: H0 (H1 is enunciated below).

5.2. Material

For the experiment, Lucane groupware platform was chosen (Fiack, 2013). It is an open source groupware platform written in Java and designed with extensibility capabilities. This platform was developed by means of the client/server architecture. It provides support for many groupware features and services, such as forum, group calendar, instant messaging, TODO-list, news feed reader or file sharing.

Despite being a groupware platform, it fails in providing appropriate awareness support. Mainly, it only features presence awareness by displaying who is connected, the ability to choose the users you want to share some events with and some awareness on who performed some specific tasks, such as posting in a forum. Therefore, it was found a good scenario to test our hypothesis (H1): improving awareness support in groupware using our framework helps in incrementing the perception about the shared knowledge that the members of a group have, which can tackle the common activities in a collaborative manner. Thus, Lucane was prepared to improve awareness capabilities so that an experiment could be conducted to test our hypothesis.

We focused on improving awareness support in the scheduling components in Lucane. Both the Calendar and TODO-List services were bound together so that the scheduling created in the TODO-List could be shared with the other users involved in the collaborative task. Thus, events are automatically created in the Calendar to reflect any change in the scheduling created in the TODO-List component. Events bound together in the Calendar show the estimated starting and ending date and when they were actually started and finished. Then, when a task is completed—by entering the actual ending date in the TODO-List—the corresponding event in the Calendar is automatically updated to show the effective completion of the task. Such updates are reflected in all the HCH interfaces of those users the task has been shared with.

Furthermore, to improve the awareness on what the other users involved in the collaborative task are doing and when they are doing so, some extra visual information is conveyed to the users. Thus, when the ending date is after the estimated date, the corresponding event is shown in red background. On the other hand, if the end date is earlier than expected, the corresponding event is shown in green background. Figure 4a reflects this aspect, which has been modified from original Lucane. The TODO-List on the upper part of Figure 4 displays two tasks to be performed by a group of users. This new feature automatically reflects both tasks in the Calendar. As Task 1 was finished before it was scheduled, it was marked in green, while Task 2, which ended later than the estimated date, is displayed in red. Corresponding arrow in Figure 4a indicates the exact part of the interface that has changed compared with original Lucane.

In order to increase collaboration even more, and also encourage group members to complete their group tasks on time, another extra functionality was included. It is a ranking of what we have called level of commitment, which measures the commitment users have in finishing their tasks on time. The level of commitment for each member is shown in the main interface once the UI is loaded, as shown in Figure 4b. This parameter is a global property inferred by the Awareness Manager component that is reflected in the particular UI for each member. Its management helps in applying some interesting considerations, such as rewarding the ranking leader with certain privileges in using the system. Both awareness features shown correspond to explicit plastic adaptations, because in both cases the information involved affects the overall group. Therefore, it requires sharing with all the teammates some awareness information by means of the server. Thus, an explicit plastic adaptation is triggered in these cases, as both the adaptation and the decision-making are delegated to the server.

Considering the example of a computer project management, we will try to emphasize how performed modifications help in improving work in group. Computer project management requires that a group of users have to work in group to achieve a series of deadlines. These deadlines may be related to deliveries of software or documentation to clients. In this sense, a tool such as Lucane provides an appropriate environment for different users to work in group. Anyhow, Lucane lacks several functionalities that would clearly improve such work in group. For instance, a correct computer project management requires assigning all necessary resources to achieve expected deadlines. In case the development team does not meet some expected deadline, the project may need to be rescheduled or more resources (such as more developers) would be required to meet the established terms. Original Lucane does not provide a way to automatically determine which deadlines have not been met, as all tasks in the TODO-List should be checked manually to determine whether a given deadline has been met or not. On the one hand, enriched Lucane graphically displays whether a deadline related to teammates has been met or not, thus being more suited to fast decision-making processes. On the other hand, the local engine (the so-called IPE) also acts, and it can be customized to automatically trigger certain actions. As aforementioned, the IPE is responsible for catching and
gathering implicitly all the events involving the work in group, as well as for triggering some particular awareness mechanisms. Thereby, we could establish that when a certain teammate finishes his/her previously scheduled task, the user who was waiting for that event is notified and, accordingly, he/she can to go on with his/her contribution (in the case of the experiment carried out, writing or completing a section in the opinion paper). These kinds of awareness mechanisms, which manage local decisions derived from the particular perception of the group, correspond to an implicit plastic adaptation.

Regarding the level of commitment, it can be used as a way to increase the motivation of the workers in the project and identify the most productive of them. These rankings can later be used to reward such employees.

Regarding the impact of applying this approach to an existing HCH interface, it is remarkable that both functionalities were included in the original platform by using exclusively the components enriched with awareness injected into the client by the IPE, and therefore in a seamless way for the underlying logic. Moreover, the particular IPE for Lucane client was derived from the generic IPF by using the IPE derivation tool, as previously explained.

As the client, the server of the distributed platform has not been modified regarding its internal functionalities. However, a new software layer was built on top of it by using Load Time Weaving to make the server act as a proxy for the Awareness Manager. Thus, the groupware platform server is unaware of the existence of this aspectual layer. The functionality of the proxy consists in intercepting all requests to the server and redirecting group-related data to the Awareness Manager. All the other information gets straightforwardly to the server.

The modified version of Lucane described will be referenced to as Tucane henceforth.

5.3. Participants
To get reliable results and conduct an experiment with people performing real groupware tasks, it was carried out by people geographically distributed in three different cities in Spain. Twelve participants took part in the experiment, and they were grouped in four groups formed by three people, each one of them from a different city. None of the participants in each group knew each other. These decisions were taken to prevent them from communicating face-to-face and thus force them to collaborate by using our system. The subjects of the experiment were between 21 and 47 years old (32 in average). There were eight men and four women.

Before the experiment, a demographic questionnaire was filled in by the participants to gather some information regarding their user profiles. This questionnaire included questions about age, previous experience with collaborative systems, and how do they perceive their usefulness. All the participants, except one, had some previous experience in using collaborative tools, and all of them agreed in the usefulness of groupware applications to improve productivity. After analyzing the results from this questionnaire, no participant was filtered out. We had
previously decided that no participant with previous experience in Lucane could take part in the experiment, which was not the case for any participant.

5.4. Questionnaires

The post-task questionnaire was carefully designed by using related works from the literature. The questions 1–4 were taken from the perceived effort questionnaire (Gutwin, 1997). The questionnaire looked at four aspects of how hard the collaborative task was: (i) overall difficulty, (ii) effort required, (iii) concentration required and (iv) difficulty of discussing the task. Questions used five-point scales with semantic anchors. Answers were translated into interval scores, using 1 to represent the greatest effort and 5 to represent the least effort. Furthermore, the questionnaire included additional questions regarding specific awareness aspects related to Who, What and When based on Gutwin and Greenberg (2002). Questions 13 and 14 were used to assess whether the participants had found useful knowing about other user’s progress or not. Finally, an open question ends the questionnaire, aiming at collecting users’ suggestions. The questionnaires were made available to the participants through an online surveying service (http://www.kwiksurveys.com).

The experiment included a final questionnaire aimed at comparing both tools. The first five questions were based on the questions proposed by Gutwin (1997) to check the preference between both groupware systems. The last three questions were aimed at providing the opinion of the participants about the new awareness functionalities added in Tucane version.

The contents of the questionnaires for both Lucane and Tucane tools and the final questionnaire are available at http://goo.gl/UoysE.

5.5. Procedure

The groups were assigned a collaborative task. This task consisted in collaboratively writing an opinion paper. To increase the motivation of the participants, two trendy topics in Spain from the time the experiment was performed were chosen for the opinion papers: Spanish Revolution as Topic 1 and Social Networks as Topic 2. To be more precise, due to the purpose of this experiment (described in Section 5.1), we asked each group to perform the collaborative task described in both versions of the tool to be compared (Lucane and Tucane), resulting in two different opinion papers for each group, each one about a different topic.

As a hint, four empty main sections (Introduction, Information, Analysis and Conclusions) were provided to the users. They had to use them as a template to collaboratively complete the article by including new subsections and the text within each of them. These and other details about the process to be followed along the collaborative task were facilitated to all of the participants sending them an instruction sheet.

Participants were required to follow the following process during writing the opinion papers: (i) to have a first contact with his/her teammates in order to decide how to divide the different parts of the article by using the QuickMessage tool (an instant messaging tool) integrated in both Lucane and Tucane; (ii) to introduce their estimated scheduling by using the TODO-List. As aforementioned, this step supports sharing the overall schedule by means of the group Calendar tool, as it has been specifically implemented in Tucane; this new awareness feature is not available in Lucane and (iii) to progressively elaborate the article by using the Forum tool integrated in both versions. Thus, teammates could work in different parts of the article in an asynchronous way, but always knowing the progress and contributions made by their teammates, consequently taking profit of a greater level of awareness, thanks to the new work in group support provided. It is worth mentioning that in Tucane the level of commitment is shown to the users when they start the session; this new awareness feature is not available in Lucane. Apart from the instruction sheet, we also provided them with a brief manual for both versions of the tool, describing how to configure the client, connect to our server and make some basic operations. To cancel out learning effect, a 2 × 2 factorial design was used so that independent variables (version of the tool and topic) were controlled in a balanced way. Thus, two groups used firstly Lucane, while the other two started with Tucane. Furthermore, we prevented that the same topic was not applied always in the same version of the tool by the different groups. Table 1 summarizes the experiment design.

After the groups finished their task with the first tool, they filled in a post-task questionnaire. Then, they used the second tool to perform again the collaborative task, but this time using the other version of the tool, and thus writing a second opinion paper. Afterwards, they filled in a second questionnaire. It is important to remark that both questionnaires were exactly the same. Finally, they were asked to fill in a final questionnaire. The questionnaires were set up in an online surveying platform.

5.6. Results Analysis and Discussion

Authors would like to point out several aspects related to the experiment itself, in order to clarify the results that were obtained and the circumstances the experiment was performed in. In this experiment, participants were given four empty sections as a template for the opinion papers, an instruction sheet and a basic procedure to follow for writing the opinion

![Table 1. Experiment design.](http://iwc.oxfordjournals.org/)
paper, just detailed in the former subsection. Apart from these indications (using the QuickMessage tool to communicate each other, the Agenda to share their estimated timing and the Forum to progressively elaborate the opinion article), any other consideration on how they should organize themselves to write down each specific part of the opinion article was included in the design of the experiment. The main reason to allow the participants to freely plan and perform the tasks with the other members of the group was to support a more natural way of collaboration, rather than a fixed fully artificial one. No analysis was made on some problems related to awareness provision, such as cognitive overload or interruptions. Authors found no reliable way to consider them in the experiment, as they would be impossible to track, given the experimentation context, and self-assessments by the participants would not be completely reliable. Regarding the cognitive overload issue, automatic cognitive load analysis techniques, such as the ones described in López-Jaquero et al., (2009), can also contribute to mitigate this problem that can appear as a result of awareness provision.

Besides, we prevented participants from using other ways to communicate each other apart from the platforms we provided them. Participants within each group did not know each other previously and had no way to communicate each other as they did not have each other’s email addresses nor telephone numbers. Furthermore, they were asked not to share any personal information among them. Although it could include some artificiality on the test, it was considered necessary, as the experiment required that the provided tools should be the only way for participants to communicate each other. Using non-controlled ways by participants to communicate each other would go against the need of correctly evaluating both tools. These kinds of limitations are common when evaluating collaborative environments (Schmidt, 2011).

The final results of the tasks given to the participants were the opinion articles they developed. Figure 5 shows the index of an opinion article completed by one of the groups. As it is displayed in the figure, the main four sections were provided as a template, and thus the administrator of the system created them. Based on this basic structure, users were able to create more subsections and fill them. Although each subsection has an owner (they are labeled with the name of the person that created it), all of them were made and filled collaboratively. The collaborative process was performed entirely by using the provided tools. Both collaborative tasks were concluded by all the groups.

Regarding the analysis of the gathered data, it is worth mentioning that only the data of 11 participants of 12 have been considered in the statistical analysis. The person whose data were left out told us that he did not read the instructions.

Figure 5. Screenshot with the content of an opinion article.
provided. Thus, in the questionnaires, he told that he had just answered the maximum value for each question. Therefore, we decided that those answers supposed a bias in the experiment and that they should not be considered. For the statistical analysis, a parametric ANOVA (ANalysis Of VAriance) test was used, as recommended in Wohlin et al. (2000). This kind of test can be used in experiments using a factorial design, as the experiment conducted in this paper. ANOVA was used to find out whether the difference between both tools was statistically significant or not.

As shown in Table 2, Tucane tool has apparently obtained better results, exhibiting a higher mean value (37.5454545 vs. 31.2727273) and a very similar standard deviation for both tools. Nevertheless, we are going to refute the null hypothesis \( H_0 \) by using the results from ANOVA (see Table 3). For an \( \alpha \) of 0.05, with \( df = 1.20 \), \( F \) must be at least 4.35124348 to reach a \( P \) value of <0.05, so \( F \) score is statistically significant. Therefore, with a \( P \) value of <0.05, we can conclude that there is a statistically significant difference between the results obtained for Lucane and Tucane. Once proved that there is a statistically significant difference between both tools, an analysis of the factors that contribute to this difference is provided by analyzing the questions and the answers gathered.

In Figure 6a, a graphical representation of the scores given by each participant to each tool is shown. The values shown in this figure represent the summation of the score given by the participant to each question in the post-task questionnaire. It illustrates how Tucane gets better results except for three participants. For two of these participants, there is no significant difference between both tools and for one Lucane is better than Tucane. Therefore, Tucane seems to be better for most participants.

Figure 6b summarizes the mean value for the answers of the participants for each question and tool for the post-task questionnaire. For the first four questions, devoted to assess the perceived effort, Tucane clearly overcomes Lucane. Nevertheless, both tools get the same mean score for the third question related to the concentration required to use the tool. In these four questions, it is interesting to remark that the second question was about the effort required to carry out a task. Tucane got a considerable better score than Lucane for this question. Therefore, participants perceived that the effort required to carry out a task was easier with Tucane, even though the task was the same for both tools.

The next three questions (5–7) were related to Who is collaborating in the system. Tucane overcomes Lucane in questions 6 and 7. Questions 6 and 7 ask about how aware the participants were of who they collaborating with and what they were doing. Although we did not explicitly add features to improve the awareness provided regarding this aspect, the perception of the participants improved according to their answers, because they were more aware of the state of the tasks and their scheduling. For question 5, they get the same result. It can be considered as natural, since this question asks the participants whether they were aware of other participants in the system. Tucane improves nothing regarding this issue; therefore, both tools should get the same score.

### Table 2. Post-task questionnaires results for mean, variance and standard deviation values.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Mean</th>
<th>Variance</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucane</td>
<td>31.2727273</td>
<td>38.8181818</td>
<td>5.940476092</td>
</tr>
<tr>
<td>Tucane</td>
<td>37.5454545</td>
<td>42.2727273</td>
<td>6.199173499</td>
</tr>
</tbody>
</table>

### Table 3. Results of post-task questionnaires for ANOVA.

<table>
<thead>
<tr>
<th>Source of variation</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>F</th>
<th>P value</th>
<th>F criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between groups</td>
<td>216.409091</td>
<td>1</td>
<td>216.409091</td>
<td>5.33744395</td>
<td>0.03166421</td>
<td>4.35124348</td>
</tr>
<tr>
<td>[1] Within groups</td>
<td>810.909091</td>
<td>20</td>
<td>40.5454545</td>
<td>40.5454545</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[2] Total</td>
<td>1027.31818</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In question 8, *Tucane* overcomes *Lucane*. This question asked about the support that both tools provide to carry out the collaborative task. Questions 9–12 are related to *When* component of awareness. Since this type of awareness is supposed to be improved in *Tucane* because of the components included, we expected better results for *Tucane*. The results confirm this hypothesis, since *Tucane* beats *Lucane* in all these questions. In question 13, there is a draw. It is also natural, because in this question, the participants were asked about their impression regarding how important they find following other participants’ progress. These results for *When* component of awareness clearly support our initial idea of how to improve this part of awareness. Finally, in question 14, *Tucane* beats *Lucane* again. This question asked about how the participant perceived the support of following the progress of the other participants. Thus, these results confirm that in *Tucane*, the participants have a better tracking of other participants’ progress.

It is worth mentioning that in the open questions in the post-task questionnaires, some of the participants who started with *Lucane* tool suggested adding some kind of method for tracking other group members’ progress, the interrelation between some services and also the visual distinction among different states of the scheduled tasks. These kinds of comments reinforce the appropriateness of the awareness support added. Also, one participant wrote down that the system should notify to the other members of the group when a participant is late or when he/she gives up the task.

Lastly, the final questionnaire was used to directly compare both tools. The results for this questionnaire are shown in Table 4. In questions 1, 2, 3 and 5, 81.82% of the participants answered that *Tucane* had better support and facilities for collaboration and was easier for work in group. Therefore, they showed a clear preference towards *Tucane*. Regarding question 4, 27.27% of the participants answered that *Tucane* required more concentration than *Lucane*. This result is interesting because it seems that improving the awareness of the tool also changed the perception of the concentration effort required to use the tool. For questions 6 and 7, a mean value of 2.82 and 3.73 (max. = 5) was obtained regarding the motivation and usefulness, respectively, of the level of commitment feature for carrying out the tasks on time, introduced in *Tucane*. Finally, with regard to question 8, a mean value of 4.45 (max. = 5) was obtained. Question 8 asks about the link between the TODO-List and the Calendar services. Almost every participant fully agreed that this link was really useful, thus showing its usefulness.

As shown in this analysis, *Tucane* clearly overcomes *Lucane*, proving that improving adaptation capabilities and work in group support in distributed collaborative environments has a positive feedback for participants, which was our hypothesis.

As it has been shown, knowing more information regarding teammates’ progress was appreciated by participants in order to help doing tasks in a collaborative manner. In short, participants perceived themselves more integrated in the working group; they could organize their work better and definitely got more motivated to elaborate the opinion paper by using *Tucane* system.

### 6. CONCLUSION

Generally speaking, there is a lack of guidelines about how to integrate awareness with all of the other aspects related to the diversity of contexts in CSCW systems. Generic HCHI does not offer an integrating and efficient solution proposal to these issues. If the goal is to encourage work in group, it is crucial to provide additional support for collaborative tasks. Awareness mechanisms can clearly contribute to a real collaboration and a deeper understanding in HCHI for multi-environment distributed scenarios. The infrastructure described in this work, based on the *Dichotomic View of Plasticity* approach, supports adding awareness capabilities to existing HCHI interfaces that do not exploit work in group concerns, independently of specific requirements and application domain. Regarding the approach, one of the most remarkable aspects is that it follows a client–server distribution model. However, as a novel factor, instead of being server-centered, it provides a balanced strategy that reduces networking dependences by providing two levels of awareness, from a local and global perspective. In this line, both engines—IPE and awareness manager—are put to work in a coordinated and engaged way along the system usage. Moreover, regarding the software infrastructure presented, we would like to remark the flexibility provided with the framework. This is observed by (i) awareness capabilities are embedded into HCHI interfaces in a completely transparent way regarding core functionalities and (ii) the instantiation process is made seamlessly. Moreover, it is independent of the underlying technology used in development. Currently, it can operate both with Java environments and .NET-based tools, considering both mobile and desktop platforms.

We have conducted an experiment to validate our approach. This experiment evaluates how participants manage a number of new work-in-group features added on *Lucane* software tool by using our approach. They were asked to answer a satisfaction questionnaire after their collaborative experience with both groupware systems: the original version of *Lucane* and *Tucane*, and the version of the tool with enriched awareness capabilities included using our approach. Participants gave a significantly better score to *Tucane*, as the mean value for the answers of most questions was better for *Tucane*. In particular, *Tucane* got better scores for 12 of 14 questions. Our hypothesis was also supported by the results for the final questionnaire, where the participants expressed a clear preference towards *Tucane*. Furthermore, the ANOVA statistical study conducted showed

### Table 4. Summary of the results of the final questionnaire.

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean (%)</td>
<td>81.82</td>
<td>81.82</td>
<td>81.82</td>
<td>27.27</td>
<td>81.82</td>
<td>2.82</td>
<td>3.73</td>
<td>4.45</td>
</tr>
</tbody>
</table>

It is worth mentioning that in the open questions in the post-task questionnaires, some of the participants who started with *Lucane* tool suggested adding some kind of method for tracking other group members’ progress, the interrelation between some services and also the visual distinction among different states of the scheduled tasks. These kinds of comments reinforce the appropriateness of the awareness support added. Also, one participant wrote down that the system should notify to the other members of the group when a participant is late or when he/she gives up the task.

Lastly, the final questionnaire was used to directly compare both tools. The results for this questionnaire are shown in Table 4. In questions 1, 2, 3 and 5, 81.82% of the participants answered that *Tucane* had better support and facilities for collaboration and was easier for work in group. Therefore, they showed a clear preference towards *Tucane*. Regarding question 4, 27.27% of the participants answered that *Tucane* required more concentration than *Lucane*. This result is interesting because it seems that improving the awareness of the tool also changed the perception of the concentration effort required to use the tool. For questions 6 and 7, a mean value of 2.82 and 3.73 (max. = 5) was obtained regarding the motivation and usefulness, respectively, of the level of commitment feature for carrying out the tasks on time, introduced in *Tucane*. Finally, with regard to question 8, a mean value of 4.45 (max. = 5) was obtained. Question 8 asks about the link between the TODO-List and the Calendar services. Almost every participant fully agreed that this link was really useful, thus showing its usefulness.

As shown in this analysis, *Tucane* clearly overcomes *Lucane*, proving that improving adaptation capabilities and work in group support in distributed collaborative environments has a positive feedback for participants, which was our hypothesis.
that the difference in the results for Tucane and Lucane was statistically significant. It is also interesting to remark how adding awareness features specially aimed at improving the perception of When facet of awareness has also indirectly improved other facets. As the results from the evaluation suggest, the perception of the concentration effort required was improved, together with the awareness of what the other participants were doing and the effort required to use the tool. To summarize, we can conclude that the results of the experiment supported our hypothesis, and thereby the presented framework has been successfully validated.

The fact that participants did not know each other was considered as a requirement for the experiment, aiming to force participants to only use the tools provided to communicate each other. As both tools have been the only way to communicate, participants have performed the overall collaborative tasks by means of provided tools, thus properly using their collaborative functionalities. In short, we have been able to provide a proper environment for the proposed evaluation. Although this fact could introduce some artificiality on the test, it was considered necessary to perform a proper evaluation. The conducted experiment does not provide an analysis on aspects such as cognitive overload or interruptions, as these aspects could not be controlled in a reliable way during the experiment. These kinds of limitations are common when evaluating collaborative environments (Schmidt, 2011).

Some of the aspects that we would like to reinforce in future implementations are the inclusion of real-time visual indications about what are the other participants doing and which interface components they are currently interacting with. Another interesting aspect is being aware of changes in the group context from session to session. These changes would be summed up and reflected in the main interface. Additionally, they could also be useful for alerting participants about important events. Open questions answered by participants in the questionnaires have confirmed the need of reinforcing such issues. Also as future work, cognitive load and interruptions during collaboration will also be considered to contribute to guide the designer of awareness features in preventing cognitive overload and unwanted interruptions.

Additionally, as Lucane client can also be executed as a web-based client, we are planning to evaluate the Lucane work in group support also in mobile devices. The inclusion of awareness components in a lighter version for mobile devices can also be instructive regarding the design of visual and interactive group-centered widgets. It is worth remarking that the infrastructure presented considers multiple purposes, including platform-related ones.

ACKNOWLEDGEMENTS

We thank the participants in the experiment and Juan José Pardo for his collaboration in the implementation of Tucane.

FUNDING

This research has been partially funded by Spanish Ministry of Science and Innovation through insPire (TIN2012-34003) grant and by Spanish Ministry of Economy and Competitiveness through InDAGuS (TIN2012-37826-C02) grant.

REFERENCES


Fuller, R. (1997) Human-computer-human interaction: how com-
puters affect interpersonal communication. In Computers, Commu-
nication and Mental Models, CRC Press Online, pp. 11–14.

conceptual framework for modeling awareness mechanisms in
collaborative systems. INTERACT’11 Proceedings of the 13th IFIP
TC 13 International Conference on Human-computer interaction—
Volume Part IV (pp. 454–457) Springer, Lisboa, Portugal.

Garrido, J.L., Noguera, M., González, M., Hurtado, M.V. and
Rodríguez, M.L. (2007) Definition and use of computation
independent models in an MDA-based groupware development

Public Devices: Making Personal Information Public, and Public

in computer-supported cooperative work-systems: structured
embedding of findings from social sciences. Int. J. Hum-Comput.
Int., 18, 323–360.

chronous groupware applications in heterogeneous environments.
In Proceedings of the 12th international conference on Groupware:
design, implementation, and use (CRIWG’06) LNCS 4154, pp.

groupware, PhD. Dissertation. University of Calgary, Calgary,
Canada.

workspace awareness for real-time groupware. Comp. Support.
Coop. W., 3, 411–446.

awareness in real-time distributed groupware: framework, widgets,
evaluation. Proceedings of HCI on People and Computers XI

workflow management. In Workshop “Towards Adaptive Workflow
cscw98/paper03/ (accessed November 2012).

for heterogeneous CSCW systems: case-study of academic
applications. IEEE International Workshop on Intelligent Data
Acquisition and Advanced Computing System: Technology and
Applications (pp. 462–467).

multiple usage paradigms. University of Michigan, Computer
Science and Engineering Division, Department of Electrical
Engineering and Computer Science.

Awareness for Memory in Education. University of Illinois, Urbana-
Champaign.

for the development of CSCW systems. In Proceedings of the 7th
international conference on Cooperative design, visualization, and
engineering (CDVE’10) (pp. 117–120). Springer, Calvia.

delivering and supporting mobile context-aware applications. J.

adaptation: user cognitive models in adaptation quality assessment.
Computer-Aided Design of User Interfaces VI (pp. 265–277)
London: Springer.

Marsic, I. (1999) DISCIPLE: a framework for multimodal collabora-

applications. In Proceedings of the 23rd International Conference
on Software Engineering (ICSE ’01) (pp. 475–484) Washington
DC, USA: IEEE Computer Society.

Myers, B. (2001) Using handhelds and PCs together. Commun. ACM,
44, 34–41.

on the World Wide Web. CSCW’96 (pp. 130–139). ACM, Boston,
USA.

Requirement-based approach for groupware environments design.

Romero-Salcedo, M., Osuna-Gómez, C., Sheremetov, L., Villa, L.,
Morales, C., Rocha, L., Ch., M. (2004) Study and analysis of
workspace awareness in C Debate: a groupware application for

Schmidt, K. (1998) Some Notes on Mutual Awareness. COTCOS-
Report. Technical University of Denmark.


A Client-Side Generic Framework for Collaborative Activities.
CRIWG 2006 (pp. 219–227) Medina del Campo, Spain: Springer.

View of plasticity to seamlessly embed accessibility and adaptivity

Delivering and Supporting Distributed Applications Enhanced with
Awareness. COMPISAC 2011. Munich, Germany.

applications through the reaction-diffusion metaphor. Comp.

to the extreme. Proceedings of the Seventh International Workshop
on Groupware, (pp. 68–73).

Component-based tailorability for CSCW applications. AI & Soc.,
14, 120–141.

awareness on the web: prototype and experiments of sharing web
page visitation information among teammates. Int. Comput., 24,
398–408.

Wessner, M., Dawabi, P. and Fernández, A. (2003) Supporting face-
to-face learning with handheld devices. In Proceedings of CSCL

