*Journal of Universal Computer Science, vol. 14, no. 9 (2008), 1447-1462*
*submitted: 30/4/07, accepted: 19/2/08, appeared: 1/5/08 © J.UCS*

# Collaborative Explicit Plasticity Framework: a Conceptual Scheme for the Generation of Plastic and Group-Aware User Interfaces

Montserrat Sendín
(GRIHO: HCI research LAB, University of Lleida, Spain
msendin@eup.udl.es)

Víctor López-Jaquero
(Laboratory of User Interfaces and Software Engineering (LoUISE)
University of Castilla-La Mancha, Ciudad Real, Spain
victor@dsi.uclm.es)

César A. Collazos
(Software Engineering Research and Development Group (IDIS)
University of Cauca, Colombia
ccollazo@unicauca.edu.co)

**Abstract:** The advent of new advances in mobile computing has changed the manner we do our daily work, even enabling us to perform collaborative activities. However, current groupware approaches do not offer an integrating and efficient solution that jointly tackles the flexibility and heterogeneity inherent to mobility as well as the *awareness* aspects intrinsic to collaborative environments. Issues related to the diversity of contexts of use are collected under the term *plasticity*. A great amount of tools have emerged offering a solution to some of these issues, although always focused on individual scenarios. We are working on reusing and specializing some already existing plasticity tools to the groupware design. The aim is to offer the benefits from plasticity and awareness jointly, trying to reach a real collaboration and a deeper understanding of multi-environment groupware scenarios. In particular, this paper presents a conceptual framework aimed at being a reference for the generation of plastic User Interfaces for collaborative environments in a systematic and comprehensive way. Starting from a previous conceptual framework for individual environments, inspired on the *model-based approach*, we introduce specific components and considerations related to groupware.

**Keywords:** plasticity, awareness, shared-knowledge, model-based approach, groupware
**Categories:** D.2.2, H.1.1, H.1.2, H.5.2

## 1 Introduction

In our current society interaction habits are quickly changing. Applications are no longer anchored to a desktop computer nor restricted to a single user. Functionalities must be supported for a wide range of hardware platforms, from desktop PCs to mobile phones, thus improving the ubiquity in interaction. Furthermore, most applications now pursue collaborative work support to make them more social and productive. Technology provides us with the freedom to not only move around while interacting with other people, but also to choose between a wide range of devices. For

instance, e-mail clients such as Google's Gmail or Microsoft Outlook target not just desktop PCs but also mobile devices. Additionally, those applications offer some collaborative features to provide an added value, such as chats or shared agendas.

This scenario introduces interesting challenges in the development of software applications, because they need to provide support for both collaboration capabilities and for heterogeneous contexts of use with different target platforms and different physical environments where interaction takes place. Since creating different versions of the application for each device and situation that can come up while interacting with an application of this kind is not a feasible option because of the high monetary and maintenance cost, devising frameworks and methods to cover their systematic development is required. Our first concern is whether groupware design is prepared to provide enough flexibility to support this kind of scenarios or not.

Some traditional [Myers 98] and current [Alarcón 06] approaches focus on the restrictions and affordances that mobility provides in groupware environments, but they do not address the huge heterogeneity of devices and the adaptation to varying constraints at the same time [Sendín 06]. Groupware solutions can be improved to provide adequate support for flexibility.

Moreover, it is well-known the importance of designing groupware taking into account not only technological issues, but also complex social dynamics where the group activity takes place. It is one of the eight challenges for groupware developers identified by Grudin in [Grudin 94]. In this sense, groupware designers have included aspects related to *awareness*, which is a cornerstone in groupware design today. Awareness reduces the meta-communicative effort needed to collaborate across physical distances in groupware environments and promotes real collaboration among group members [Palfreyman 96]. However, despite the necessity of a clear overall picture of it, awareness support is not systematic and developers must rebuild it from scratch for each new system. These kinds of issues need sophisticated capturing and processing capabilities, the so-called *awareness mechanisms*, whose integration in the development of collaborative systems is not an easy task. It is not until recent years that awareness support has begun to blunt from the CSCW (Computer Supported Collaborative Work) community.

Under the lack of solid guidelines regarding awareness we present our second concern: the integration of awareness in groupware development in a systematic and comprehensive way.

In order to offer enough flexibility to accommodate a specific system to different devices and situations, it is necessary to consider the context under a wide perspective. In a broad perspective, the variability and multiplicity of parameters introduced by all the previous issues (mobility, heterogeneity and adaptation to changing contextual situations) are gathered under the *plasticity* term. It was coined in 1999 as the ability of systems to mould their own User Interface (UI henceforth) to a range of computational devices, conditions and environments in order to tackle the diversity of contexts of use in an economical and ergonomic way [Thévenin 99]. Since then, a great amount of tools have emerged offering a solution to some of these issues. Although plasticity tools have focused on individual scenarios, their tools can be revised and specialized for novel groupware scenarios, where issues related to mobility and heterogeneity appear.

Our work is in the line of reusing some advanced work in the field of plasticity in order to integrate and exploit awareness information as an integral part of the plasticity process. The aim is to provide flexibility and awareness in a comprehensive and systematic way.

In this sense, we introduce the following three contributions: (1) The incorporation of *group-awareness* in the characterization of the context of use, becoming one more parameter to be embedded into the plasticity process and thus enriching both the contextual information and the subsequent adaptation. (2) The integration of specific *awareness mechanisms* both in the client and in the server-side of the plasticity process. Finally, as in a groupware environment it is essential that each member shares his/her individual understanding with the group [Ellis 91], (3) the integration of a global view of the *shared-knowledge* is fundamental to be gathered and handled on the server-side of the infrastructure. By *shared-knowledge* we understand the knowledge that embraces all those aspects of the collaborative work that could be useful in the development of the group activity, from a global perspective. It provides a shared understanding of the problem that allows maintaining the consistency of the *awareness* information as well as inferring overall properties in order to enrich the individual perception of each member.

According to our *Dichotomic View of plasticity* [Sendín 04], there are two types of plasticity: implicit and explicit. When the complexity involved in the adaptation process is too high to be handled on the client-side, the client will request the server for an *explicit plastic* adaptation, which implies the generation or reconfiguration of a new UI. On the other hand, if the adaptation complexity is low enough, the client will tackle it, providing an *implicit plastic* adaptation.

Since the integration of *awareness mechanisms* in each target device (addressed to promote user-to-user interactions and to construct individual understandings about the group) has already been tackled in [Sendín 06], the paper is focused in the *explicit* part of the framework. In particular, a conceptual reference framework is presented for the construction of tools to generate plastic UIs, offering support for collaborative work. It consists of a framework built on the *model-based approach* that structures the problem space and provides the foundations for a solution using high level specifications. The aim is to provide an instrument for the study, comparison and analysis of these kinds of tools. The framework is called *Collaborative Explicit Plasticity Framework* (Collaborative EPF henceforth).

The paper is structured as follows. Section 2 discusses some related work. Sections 3 presents our conceptual framework for the development of plastic and group-aware UIs proposed. Then, we present AB-UIDE, a tool that implements all of the guidelines gathered in the Collaborative EPF except the ones related to groupware, which is part of our further work. A brief discussion of the possible benefits, some conclusions and further work conclude the paper.

## 2 Related Work

The path towards ubiquitous computing, the growing complexity of user interfaces and the wide diversity of potential users of the applications have promoted the development of systems able to adapt to the context of use they are executed in. The development of user interfaces with some adaptation capabilities is not a new issue

[Benyon 93]. Adaptation has been an active research field for the last twenty years, especially to address issues related with intelligent tutoring systems in adaptive hypermedia. Techniques such as adaptive link ordering or hiding [Brusilovsky 01] have been extensively used in this kind of systems. In the field of *model-based* user interfaces development [Paternò 99] there have been different initiatives aimed at providing adaptation facilities to support multi-platform applications [Eisenstein 00] too. Nevertheless, adaptation per se is not always necessarily good. It is necessary to also preserve usability trying to pursue plasticity following the motto *"specify one UI usable, generate multiple"* [Thévenin 99].

The most relevant work on the field of plasticity is the one developed by the IIHM group from Joseph Fourier University. In their original plasticity framework, the context of use was limited to the running platform and the physical environment [Thévenin 99], leaving out the user. In [Calvary 00] a new development process model is presented to supply the lack of appropriate notations related to context of use. The process is defined as a combination of vertical reifications and horizontal translations. In [Calvary 01a] they present ARTStudio, a concrete but incomplete implementation of the framework.

In [Calvary 02a] the initial framework is revised introducing a more general process that refines the design time process and extends its coverage to the runtime, according to the software mechanism and process proposed in [Calvary 01b]. In relation to the design time, it introduces reverse engineering, multiple entry points (a multi-path development [Limbourg 04]) and a cross operator at the translation level. The CAMELEON Reference Framework is the next release [Calvary 02b] in which ontological models are introduced and defined as meta-models that can be instantiated into archetypal (design process) and/or observed (executable) models. They serve as the central source of knowledge in the framework. These models are redefined in [Calvary 03] based on the distinction between *predictive* –foreseen at design time– and *effective* –foreseen at runtime– contexts of use. Furthermore, a new attribute is incorporated to the context of use: the user.

In [Calvary 04] plasticity definition was revised to take into consideration a well-known fact: adaptation can impact not just the presentation layer, but also the functional core. Based on that, a new reference framework is presented to cover these issues, which is applied at widget granularity. As a result, they provide the notion of a Comet (COntext sensitive Multi-target widgETS), trying to make the functional equivalence of widgets explicit. They define a Comet as an introspective interactor that publishes the quality in use it guarantee for a set of context of uses, the user tasks and domain concepts they are able to support, as well as the extent to which they support adaptation. Thus, with Comets, plasticity is expressed directly within the widgets, instead of the interactive system, going from coarse grain into finer grain.

These plasticity frameworks, arisen from a successive evolution, serve well as the basis for the development of plasticity tools for individual scenarios. Nevertheless, some key issues remain open: (1) the plasticity framework described so far does not consider the current task the user is carrying out in the context of use, introducing limitations in the possible adaptations supported by the framework; (2) it does not integrate enough directives or strategies to preserve usability; (3) as aforementioned, it does not consider the intrinsic aspects to collaborative environments such as *awareness* and any kind of *awareness mechanism*; and finally, (4) it does not consider

our *dichotomic view of plasticity*, what means that the framework tackles both predictive and effective contexts of use.

As frameworks that specifically consider group issues, we can mention the Brézillon et al. work [Brézillon 04]. They have developed a study analyzing different groupware systems taking into account the context awareness point of view. They have analyzed three applications that although do not support context in an explicit way, they use several contextual aspects and elements in order to support group work. They have developed a framework that includes group work as a knowledge processing task with some machine supporting activities, identifying different types of contexts at different levels of generality.

In short, there are many projects trying to integrate different aspects in the groupware design in order to support collaborative activities considering the context. However, there is a lack of guidelines about how to integrate all of the aspects aforementioned in the same tool. With a few exceptions, awareness support presented up to now involves localized solutions to specific domain problems, as well as isolated approaches and principles that are difficult to generalize to other situations.

## 3 Conceptual Framework for the Server-side

The infrastructure to be placed in the *plasticity server* consists in a systematic development support capable of generating or reconfiguring a suitable UI for each new contextual situation not solvable on the client-side. We call this kind of tools, which follow a *model-based* approach, *Explicit Plasticity Engines* (EPE henceforth). In this section we explain the ideas that lay the foundations of the conceptual framework and that have been defined aimed at being a reference for the construction of concrete EPEs, in which, for the first time, we introduce the considerations about the group. The conceptual framework defined is called *Collaborative EPF*.

### 3.1 Foundations of our Conceptual Framework: the Collaborative EPF

With the aim of systematizing the construction of plastic UIs, it is necessary to adopt the motto "write once and execute everywhere, anytime". The main idea is to specify a unique, generic and abstract UI, flexible enough to tackle multiple variations. We are making reference to the *Abstract UI*, defined as a high-level and platform and modality independent specification that allows describing the UI by means of high level elements. In that sense, *model-based* methods [Paternò 99] provide a mechanism to design the UI by means of a number of declarative models that describe not only the static and dynamic aspects of the UI, but also all its relevant factors, which include the different requirements of every context of use. Then, these declarative models are translated into some code directly executable on a specific platform or into some kind of intermediate language (usually an XML-based language), which can be interpreted by a renderer. In short, *model-based* methods integrate the knowledge of the UI into a UI design method, giving the required formalisms to build UIs in a systematic way.

The framework presented on this paper makes up a revision from a previous version (the *Explicit Plasticity Framework* –EPF), which did not include any information about the group [Sendín 05]. Thus, the framework revised specifies not

only adaptation capabilities, but also collaboration issues. The key components to incorporate in an EPE in order to integrate the group issues in the underlying plasticity process are the following ones: (1) the inclusion of the *shared-knowledge*, to be represented by a model not included so far: the group model (GroupM henceforth); (2) the specification of the restrictions and requirements related to a particular collaborative system (the *collaborative rules* component); and (3) other recognized groupware guidelines gathered from experience (the *collaborative guidelines* component).

The framework fosters the separation of concerns usually found in *model-based approaches* with a progressive multi-layer design of the UI. Hence, it is structured in four levels of abstraction [Calvary 03]. From the most abstract to the most concrete they are: (1) a representation of the tasks involved (a task model), the domain concepts (represented in a domain model) and the human-computer conversation (the dialogue model). The task model and the domain model are specified manually by the UI designer. The dialogue model can be either derived automatically from a detailed task model, or it can be specified manually; (2) the *Abstract UI*, defined above; (3) the *Concrete UI* (a concrete instance detailed enough from the *Abstract UI*, which is modality-dependent and platform-independent); and (4) the *Final UI* (a UI implemented in the final code particular to each target platform, ready for the linkage step or to be interpreted; it is the actual UI the user will interact with).

As in our previous framework, a *shared model approach* [Griffiths 99] is followed, which consists in supporting the UIs generation process by means of the use of *model* repositories, understood as common areas where models contribute and share concepts, and where the opportune restrictions are applied. Figure 1 depicts our conceptual framework revised, including thereby the groupware components.

In the following subsections the models and other components implied, the phases that compose the framework, some remarks on the activation of the engine, and finally some considerations about the group are introduced. We follow the order in the explanation we consider the most adequate to understand the engine as a whole.

### 3.1.1 Models involved in the Framework

Although there is no standard to define the models needed in order to specify a UI, there are some of them widely used, such as the task model, the domain model, the user model and the dialogue model. The models that we consider relevant for the purpose described, as well as the implication that each of them has in the development of plastic and group-aware UIs are the following ones:

- Task model (TaskM): structured representation of the tasks a user can carry out through the UI.
- Domain model (DomainM): represents the objects the user task interacts with.
- User model (UserM): representation of features and aspects related to the user, such as the knowledge level, preferences, goals, state, needs, etc.
- Dialogue model (DialogueM): it is used to describe the human-computer conversation, that is, the “dialog” between the UI and the user. The dialogue model allows establishing the style of navigation.
- Spatial model (SpatialM): detailed spatial description from the real world, which is in charge of providing location-awareness.

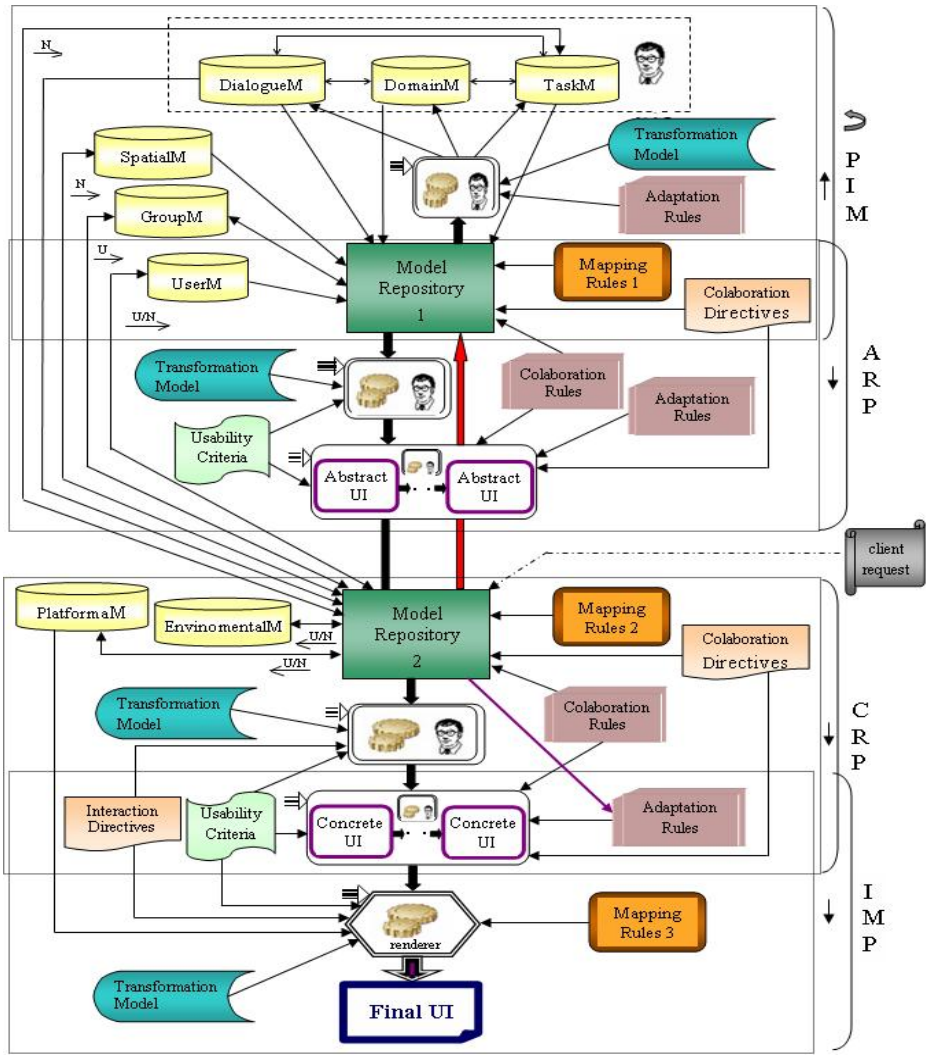



*Figure 1: Conceptual Framework for UIs generation (Collaborative EPF) Sketching*

- Platform model (PlatformM): explicit expression of the target platforms in terms of quantified physical resources and software features such as the operating system, the version of the java virtual machine, the configuration, and so on.
- Environmental model (EnvironmentalM): consideration of any temporal (the time, the day of the week), or environmental conditions (such as the daylight, the level of ambient noise or the weather conditions), which are specific for each domain of application and that also influence in the adaptation.
- Group model (GroupM): a representation of the *shared-knowledge*. We are referring to a certain kind of common repository or working group memory,

which is also to be modelled in an XML-based language. It must be conceived as a collection of the individual perceptions (the *particular group-awareness*) of every group member, as far as this information is being communicated to the server via request.

### 3.1.2 Phases that comprise the Framework

Following the structure in four levels of abstraction mentioned above, the framework is composed of three sequential phases, which operate in this order: (1) *Abstract Rendering Process* (ARP in figure 1); (2) *Concrete Rendering Process* (CRP); and (3) *Implementation* (IMP).

The *Abstract Rendering Process* derives the *Abstract UI* from the TaskM and the DomainM, intervening also these other models: SpatialM, UserM, DialogueM and GroupM. Then, the *Concrete Rendering Process* obtains the *Concrete UI* from the *Abstract UI*, managing the selection of the set of *Concrete Interaction Objects* (CIOs) [Vanderdonckt 93], according to all the contextual information represented in the following models: UserM, SpatialM, PlatformM, EnvironmentalM, GroupM, and governed by the DialogueM. For instance, from an *Abstract UI*, it could be derived either a textual, graphical or speech UI, according to all the contextual information (the user preferences, the level of ambient noise and the device).

More specifically, this phase consists in deciding which concrete component of the UI will represent the functionality described by each abstract component. The goal is to progressively shape the suitable UI for each case, taking into account the spatial and contextual circumstances, the target platform, the final user and the group constraints. The expected *Concrete UI* results from the restrictions propagation regarding the contextual issues. This process of transformation must be led towards maximizing usability. This is the purpose of the *interaction guidelines* component.

The third phase –IMP in figure 1- generates the *Final UI* from the *Concrete UI*. It is carried out by the rendering engine –renderer in Figure 1- that automatically transforms the visual aspect and dynamic behaviour of the *Concrete UI* into interpretable or executable code, by replacing each CIO or set of CIOs with a final widget or set of widgets to take part in the *Final UI* code. In particular, a group of mapping rules is in charge of translating the CIOs to the final widgets according to the platform specification (the PlatformM).

Depending on the scope of the adaptation and on the involved concepts handled, in some situations it is enough to activate phase 2, without readjusting the *AbstractUI*, and in other situations it is necessary to process both phases. Each group of automatic tools has an *entry point*[1] reflected by a double floating arrow on the left side. The red ascendant arrow that connects both *model repositories* in Figure 1 indicates the feedback related to the context of use between them. Sometimes it is enough to translate a certain UI to another language by using the appropriate renderers. The entry point in the *Implementation* phase represents this possibility.

[1] A possible initial point in the process of exploitation of models.

### 3.1.3 Additional Framework Components

Apart from the models mentioned, the framework is composed by these components:

- *Mapping rules.* These formalize the relationships and restrictions between the models that are necessary to accomplish both phases. A set of rules is required for each one of the phases to manage the relations in each group of models.
- *Model repositories*. A model repository -common area- for each phase is required, making it possible to share concepts among each set of models. The mapping rules act upon the repositories to obtain the successive refinements of the UI following a constraint propagation process.
- *Adaptation rules*. These are rules applicable to the *Abstract UI*, the *Concrete UI* and even the models from the first level of abstraction to adapt them to a new contextual situation, avoiding starting from scratch the whole generation process of each model every time. Even more, these rules could also modify themselves in order to reflect the refinement of an adaptation rule according to new inferred data.
- *Transformation model.* It describes the transformation process from an *Abstract UI* into a *Concrete UI*. It determines which CIOs will represent each set of *Abstract Interaction Object* (AIO) [Vanderdonckt 93].
- *Usability criteria*. They define the usability requirements of the application at hand. They also restrict the possible evolution of the UI and establish which usability criteria should prevail when the adaptation engine must choose between conflicting adaptations, avoiding usability degradation. To preserve the usability of the generated adapted UIs as much as possible, the system checks if the usability properties in the adapted UI improve those in the original one. Usability criteria must be particularized for each platform and system [López-Jaquero 05]. These criteria are taken into account in all the three phases.
- *Interaction guidelines.* These are any type of style guidelines, ergonomic heuristics or usability patterns that gather the experience from UI designers, making their reutilization possible. They guide the process of transformation of models, according to the contextual restrictions, in order to preserve usability.
- *Collaboration rules.* They are specific rules that govern the collaborative behaviour of the system at hand from a global perspective. These are the rules applied when the server receives a request by a group member, containing collaborative issues (pieces of information of interest for the whole group activity). They determine either whether it is necessary to activate the process to generate a new *Abstract UI* –either by adapting it from a previous one applying the *adaptation rules*, or deriving a new UI from the fist level of abstraction-; or, if it is enough, registering that event in the GroupM. Additionally, they provide mechanisms to infer global properties from the *shared-knowledge* in order to support decision-making upon specific situations. Naturally, these rules are particular to each system. They intervene in the first two phases.
- *Collaboration guidelines.* They define some guidelines that gather the experience accumulated by groupware researchers, trying to be useful to promote participation between group members, thus contributing to a real collaboration. One of the most important is the integration of diverse *global awareness mechanisms*. By *global awareness mechanism* we are referring to any mechanism

destined to capture, maintain, update and increment the *shared-knowledge*, as well as its distribution to the entire group members, aiming at contributing and promoting a real collaboration and a major effectiveness in the development of the group activity. These guidelines intervene in the first two phases, in order to drive how the *shared-knowledge* can be taken into account properly on the modelling of a UI from scratch.

### 3.1.4 Specific considerations about the group

The TaskM has to be especially accommodated to a group scenario. This implies including not only specific tasks about the group activity, but also little actions aimed at promoting collaboration in any type of task. Furthermore, apart from describing the usual issues in a TaskM, it is required now to specify the individual or collaborative features of each task. Moreover, the relationship between the TaskM and the GroupM takes a key relevance in a group activity. In this sense, it is firmly recommended to specify explicitly the relation between both models in order to guarantee that the working group situation and the state of the group activity (the 'group snapshot' henceforth) have the appropriate relevance in the UIs construction process. These relationships are specified between the *Mapping Rules 1* (corresponding to the *Abstract Rendering Process* phase) and the *collaboration rules*.

In this sense, the conceptual framework includes some heuristics and recommendations promoting the relevance of specifying as much detailed as possible (a) the relations between the tasks –TaskM- and the group snapshot –GroupM-; and (b) the transition between states of the UI –DialogueM- and the group snapshot –GroupM-, in order to derive UIs the most conforming to group situation as possible, that is, group-aware UIs.

### 3.1.5 Activation of the Framework

Under our *Dichotomic View of plasticity*, an EPE acts initially to produce the initial UI and it is not reactivated until receiving a client request with the current contextual restrictions, triggering again the process. This is depicted on Figure 1 right side.

The use of model repositories, in particular *model repository 2*, contributes to the mechanism of propagation of any change produced in the UI or in the context of use during the interaction (connection with the 'client request' in Figure 1). In fact, this is the last link in the chain of the propagation of contextual changes to be notified previous to the process of generation of a new specific UI. Precisely, most of the *model-based* tools in the literature ignore this step.

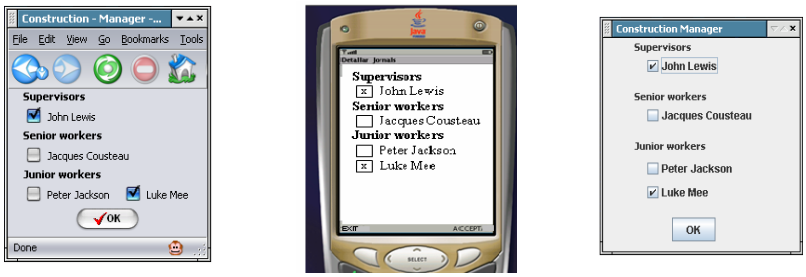
Hence, each model is either updated (as in the case of the GroupM), or simply notified (upon a new user's location –SpatialM- or a new task to carry out –TaskM). Depending on the cases, the models intrinsically contextual may require (a) an updating in order to reflect dynamic changes in the actual context of use, or (b) a notification reflecting a total replacement. In particular, a notification would reflect situations such as a change in the person who is using the system –UserM-, a radical change in the environment –EnvironmentalM-, or a migration between devices –PlatformM. Labels 'U' for *Updating*; 'N' for *Notification*; and 'U/N' for this double possibility indicate each situation in Figure 1. In some cases (e.g. in the notification of a new device), a consequent adaptation, extension or pruning operation in the TaskM

or even in the other high-level models are promoted. This is reflected by the connection between the automatic tools on the top of *model repository 1* and these models. Because these adaptations can make up one more step in the process, it can be considered as a new phase: *Preparation of the Initial Models* –PIM in Figure 1.

## 4 AB-UIDE: an EPE that Supports our Framework

AB-UIDE [López-Jaquero 05] is a particular EPE that supports our conceptual framework. All its components are implemented except for those related to groupware. Next, the most relevant techniques and languages used are presented.

- *Task Model.* It uses a combination of three different notations: state diagrams, CTT [Paternò 99] and the abstract interaction tools [Constantine, 03].
- *Abstract UI.* The AIOs proposed for UMLi in [Pinheiro 02] are used, enriched by a new type of AIO: the *selector*.
- *Concrete UI.* It uses the *Concrete UI* model proposed for UsiXML.
- *Final UI production.* It is generated by means of *renderers*. The available ones so far support the generation of the UI for Java (the J2SE modality), Java for mobile devices (J2ME) and XUL+Javascript.
- *Usability criteria specification*. In AB-UIDE this component is called *usability trade-off*. It describes the weight and influence that each usability criterion must have in the generation of the UI. To capture the usability requirements a variant of the *Goal-oriented Requirement Language*[2] is used.




(a) (b) (c)

*Figure 2: FUI generated by AB-UIDE for XUL (a), J2ME (b) and J2SE (c).*

All the models in AB-UIDE are stored in a language based on UsiXML [Limbourg 04], a UI description language based on XML, which also supports all the transformation process between models or between different versions of a model [Limbourg 04]. Figure 2 shows a concrete screenshot of a *Final UI* generated for XUL (a), J2ME (b) and J2SE (c) respectively.

[2] http://www.cs.toronto.edu/km/GRL/

# 5 Discussion

It is obvious that an effort must be done in order to provide flexible and integrating solutions in groupware design. We divide this discussion in three blocks in order to identify our contributions in a better way.

## 5.1 Collaboration and awareness

In a collaborative environment *awareness mechanisms* should be provided to share each *particular group-awareness* with the group members in order to assure a high interaction and whole *shared-knowledge awareness* [Gutwin 02]. The fact that the group issue is integrated as one more parameter to be embedded into the plasticity process, as well as the mechanism of propagation provided by an infrastructure based on the *dichotomic view of plasticity* offer both a means to share the context.

Furthermore, this sharing mechanism follows a client-server distribution approach that instead of being server-centred, it tries to provide a balanced strategy in the line of obtaining the advantages from both a server-centred and fully distributed approaches found in the literature. In fact, under the *dichotomic view of plasticity*, awareness information is maintained on both sides (client and server), looking for an operational balance. Precisely, the *shared-knowledge* is conformed by the compilation of each *particular group-awareness*. In short, the idea is to combine both perspectives of awareness in the most convenient way, resulting in a versatile combination of two levels of awareness: (1) a distributed *shared-knowledge* that promotes real time communication and coordination between group members, contributing to autonomy and robustness at the client side; (2) a centralized *shared-knowledge* in the server. In short, this view is in the line of obtaining a trade-off between the degree of awareness and the network usage, fostered by Correa and Marsic [Correa 03].

## 5.2 Flexibility and systematization

The conceptual framework presented in this paper includes all the aspects regarding heterogeneity of devices and diversity in physical environments within an integrating framework that also focuses on the needs and characteristics of collaborative environments. Our approach follows a context-centred approach that pursues an understanding of the prospective users' workplace characterizing and modelling each scenario by its contextual and particular constraints.

Regarding the systematization issue, we must keep in mind that the essence of plasticity is to tackle the diversity of contexts of use in an economical and ergonomic way. In fact, *model-based approaches* cover the systematic development of plastic UIs for different contexts of use and provide the foundations for code generation, what contributes to reutilization and cost of production alleviation.

## 5.3 Originality in the conception

The conceptual framework presented is an extension to the CAMELEON Reference Framework [Calvary 02b], in the different ways exposed next. It has two main goals: (1) to provide an instrument for the study, comparison and analysis of these kinds of tools; and (2) to guide in the process of producing UIs characterized not only for

being plastic, but also group-aware, that is to say, UIs customized to the evolving working group conditions. In this sense, it is essential to handle an overall perspective of the *shared-knowledge* in order to be opportunely exploited along the derivation of models. In fact, only deploying group-aware UIs is possible to spread the so-called *shared-knowledge awareness* [Collazos 02], so that a global context for the group activity can be assimilated by all the group members, which can then work in a collaborative manner.

Apart from the considerations of the group, it offers a complete specification of the rules, criteria, directives and knowledge bases that participate in the process, showing their dependencies. This level of detail allows a better understanding of the method and development process followed in the tool being object of study. The CAMELEON reference framework does not allow analyzing these questions in depth. The inclusion of heuristics and recommendations in the Collaborative EPF is also in the line of offering a deep orientation in the specification and exploitation of models.

Regarding the models, the Collaborative EPF explicitly introduces a new model – the GroupM- and reinforces the use of other models not employed in a generalised way in the literature, promoting as well the dialogue model in order to obtain a more detailed description of the navigation style and the dynamic behaviour of the UI.

Other contributions regarding the CAMELEON Reference Framework are the use of metrics in the evaluation of the *usability criteria* component, making the process a use-centred design. In fact, this component, devised for the preservation of a set of predefined and system-specific usability goals, makes up the artefact that distinguishes plastic UIs support from multi-contextual UIs support.

The consideration and integration of the current task in the process of derivation of UIs is another of our extensions. In effect, the task at hand makes up the key piece of information in every moment. Finally, the Collaborative EPF is specially adapted, but not limited, to the application of the *dychotomic view of plasticity*, which implies (1) the necessity of a mechanism of propagation and anticipation of contextual changes and (2) the delegation of the proactive adaptations to contextual changes to the client.

## 6 Conclusions and Future Work

Assuming that mobile solutions have gone beyond the role of personal tools to offer large-scale solutions in supporting coordinated work, it is recommendable to reuse as far as possible the work already realized to solve problems inherent to mobility in the groupware work. In this sense, the plasticity field tackles the diversity of contexts of use offering a solid and experienced support to flexible, systematic, reusable and low-cost production of highly mouldable UIs. Our work is in the line of embedding group implications as one more element to be contemplated on the *model-based* machinery, looking for a seamless amalgam between plasticity and awareness. We have devised the theoretical foundations to do so in a concrete plasticity tool giving rise to a comprehensive conceptual framework that extends previous proposals as much in the group as in other issues. The tool chosen is framed in an integrating infrastructure that also supports *implicit plastic* adaptations, providing another level of awareness. The two levels of awareness fostered –centralized and distributed– match the two levels of

plasticity promoted by our *dichotomic view*, providing a valid mechanism to build and spread *shared-knowledge awareness* via the UIs produced.

As further work, we are preparing to put in practice the guidelines presented in this paper in the AB-UIDE tool, instantiating each group component in a particular scheme, as a first implementation of the Collaborative EPF. Additionally, we would like to focus our long-term work on (1) incorporating mechanisms to make it easier to make decisions about what the best appliance for each workplace is, allowing the user alternating between either one making up his/her mind, or being suggested by the system; and (2) identifying some kind of usability directive in which the collaborative work is taken into account explicitly.

**Acknowledgments**

This work is partly supported by the Spanish project ADACO: ADAptive and COllaborative web based systems (TIN2004-08000-C03-01).

## References

[Alarcón 06] Alarcón, R.A., Guerrero, L.A., Ochoa, S.F., Pino, J.A.: “Analysis and Design of Mobile Collaborative Applications using Contextual Elements”; Computing and Informatics, 25, 6 (2006), 469-496.

[Benyon 93] Benyon, D. and Murray, D.: “Developing adaptive systems to fit individual aptitudes”; Proc. IUI, ACM Press, Orlando, U.SA. (1993), 115-121.

[Brézillon 04] Brézillon, P., Borges, M., Pino, J., Pomerol, J-C.: “Context-awareness in group work: 3 case studies”; Proc. IFIP Conf. on Decision Support Systems (2004), 115-124.

[Brusilovsly 01] Brusilovsky, P.: "Adaptive Hypermedia"; UMUAI (User Modeling and User-Adapted Interaction), 11 (2001), 87-110.

[Calvary 00] Calvary, G., Coutaz, J., Thévenin, D.: “Embedding plasticity in the development process of interactive systems”; Proc. 6th Workshop *User Interfaces for All*, Bristol (2000).

[Calvary 01a] Calvary, G., Coutaz, J., Thévenin, D.: “A Unifying Reference Framework for the Development of Plastic User Interfaces”; Proc. IFIP WG2.7 (13.2) Working Conference EHCI'2001, Springer Verlag, LNCS 2254, Toronto (2001), 173-192.

[Calvary 01b] Calvary, G., Coutaz, J., Thévenin, D.: “Supporting Context Changes for Plastic User Interfaces: A Process and a Mechanism”; Proc. Human-Computer Interaction IHM-HCI. A. Blandford, J. Vanderdonckt and Ph. Gray (eds.), Vol. 1, Springer-Verlag (2001), 349-363.

[Calvary 02a] Calvary, G., Coutaz, J., Thévenin, D., Limbourg, Q., Souchon, N., Bouillon, L., Florins, M., Vanderdonckt, J.: “Plasticity of user interfaces: a revised reference framework”; Proc. TAMODIA'2002, Bucharest (2002), 127-134.

[Calvary 02b] Calvary, G., Coutaz, J., Thévenin, D., Bouillon, L., Florins, M., Limbourg, Q., Souchon, N., Vanderdonckt, J., Marucci, L., Parernò, F., Santoro, C.: “The CAMELEON Reference Framework”, Deliverable D1.1, September 3th, (2002).

[Calvary 03] Calvary, G., Coutaz, J., Thévenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: “A Unifying Reference Framework for Multi-Target User Interfaces”; Interacting with Computers, Elsevier Science B.V., 15, 3 (2003), 289–308.

[Calvary 04] Calvary, G., Coutaz, J., Dâassi, O., Balme, L., Demeure, A.: "Comet: a new generation of widget for supporting user interface plasticity"; Proc. EHCI/DSV-IS'2004, LNCS, Springer-Verlag, 3425 Hamburg (2005), 306-324.

[Collazos 02] Collazos, C., Guerrero, L., Pino, J., Ochoa, S.: "Introducing Shared-Knowledge-Awareness"; Proc. Information and Knowledge Sharing, St. Thomas, (2002), 13-18.

[Constantine 03] Constantine, L.L.: "Canonical Abstract Prototypes for Abstract Visual and Interaction Design"; Proc. DSV-IS 2003, LNCS, Springer-Verlag, 2844, Madeira (2003), 1-15.

[Correa 03] Correa, C., Marsic, I.: "A Flexible Architecture to Support Awareness in Heterogeneous Collaborative Environments"; Proc. CTS 2003, Orlando (2003), 69–77.

[Eisenstein 00] Eisenstein, J., Vanderdonckt, J., Puerta, A.: "Adapting to Mobile Contexts with User-Interface Modeling"; Proc. IEEE Workshop on Mobile Computing Systems and Applications, English Lake District (2000), 83-92.

[Ellis 91] Ellis, C.A., Gibbs, S.J., Rein, G.L.: "Groupware: some issues and experiences"; Communications of the ACM, 34, 1 (1991), 38-58.

[Griffiths 99] Griffiths, T., et al.: "Teallach: A Model-Based User Interface Development Environment for Object Databases"; Proc. UIs to Data Intensive Systems (1999), 86-96.

[Grudin 94] Grudin, J.: "Groupware and Social Dynamics: Eight Challenges for Developers"; Communications of the ACM, 37, 1 (1994), 92-105.

[Gutwin 02] Gutwin, C., Greenberg, S.: "A Descriptive Framework of Workspace Awareness for Real-Time Groupware"; Computer Supported Cooperative Work, 11, 3-4 (2002), 411-446.

[Limbourg 04] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: "UsiXML: a Language Supporting Multi-Path Development of User Interfaces"; Proc. EHCI/DSV-IS'2004, LNCS, Springer-Verlag, 3425 Hamburg (2005), 200-220.

[López-Jaquero 05] López-Jaquero, V.: "Interfaces de Usuario Adaptativas Basadas en Modelos y Agentes Software"; PhD Thesis. University of Castilla-La Mancha (2005).

[Myers 98] Myers, B., Stiel, H., Gargiulo, R.: "Collaboration using multiple PDAs connected to a PC"; Proc. ACM CSCW'98 Seattle (1998), 285-294.

[Palfreyman 96] Palfreyman, K.A., Rodden, T.: "A protocol for user awareness on the World Wide Web"; Proc. CSCW'96, Boston, (1996), 130-139.

[Paternò 99] Paternò, F.: "Model-Based Design and Evaluation of Interactive Applications": Springer Verlag (1999).

[Pinheiro 02] Pinheiro da Silva, P.: "Object Modelling of Interactive Systems: The UML*i* Approach"; Ph.D. Thesis, University of Manchester (2002).

[Sendín 04] Sendín, M., Lorés, J.: "Plasticity in Mobile Devices: a Dichotomic and Semantic View"; Workshop on Engineering Adaptive Web, supported by AH 2004, (2004), 58-67.

[Sendín 05] Sendín, M., Lorés, J.: "Remote Support to Plastic User Interfaces: a Semantic View"; *Selection of HCI related papers of Interacción 2004*. Springer-Verlag (2005), 55-70.

[Sendín 06] Sendín, M., Collazos, C.A.: "Implicit Plasticity Framework: A Client-Side Generic Framework for Collaborative Activities"; Proc. CRIWG 2006, LNCS, Springer Verlag, 4154 (2006), 219-227.

[Thévenin 99] Thévenin, D., Coutaz, J.: "Plasticity of User Interfaces: Framework and Research Agenda"; Proc. Interact'99, IFIP IOS Press Publ., (1999), 110-117.

[Vanderdonckt 93] Vanderdonckt, J., Bodart, F.: “Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection”; Proc.ACM Interchi’93.ACM Press, (1993), 424-442.