Implicit Plasticity Framework: A Client-Side Generic Framework for Collaborative Activities

Montserrat Sendín[1] and César A. Collazos[1,2]

[1] GRIHO: HCI research group
University of Lleida, 69, Jaume II St., 25001- Lleida, Spain
msendin@eps.udl.es
[2] Department of Systems, FIET
University of Cauca, Campus Tulcan, Popayán, Colombia
ccollazo@unicauca.edu.co

Abstract. We are interested in integrating and exploiting the *shared-knowledge* from a group by an existing infrastructure of plasticity, as another parameter more to be embedded in the adaptation process. The aim is to offer the benefits from *plasticity* and *awareness* jointly, providing a systematic support in both issues. In this paper we focus on the proactive adaptation to contexts of use under a plasticity viewpoint. The aim is to promote interaction and real time coordination, contributing to real collaboration in multiple and changing groupware scenarios.

1 Introduction

We are no longer tied to our desktop computer due to the wireless technology, the mobile networking capabilities, and a plethora of new computing technologies. New advances provide us freedom to move around and to access to the technology in new and changing environments, keeping in permanent contact when we are working on groups. However, current CSCW approaches focus on the restrictions and affordances that mobile devices and mobility provide, but they do not address the huge heterogeneity[1] and the adaptation to changing contexts of use at the same time. Real time constraints related not only to the shared-knowledge between group members, but also other related to the user (changing needs and preferences), to the environment (daylight, localization, etc.), and even related to network constraints (bandwidth, server availability), which describe the *context of use*[2], are volatile and require sophisticated capturing and adaptive capabilities that today are still challenging. In a broad perspective, the variability and multiplicity of parameters introduced by all the previous issues are collected under the term *plasticity*. It was coined as the ability from systems

[1] Diversity and versatility in the decision about the device or platform to use, taking into account the significant differences in their physical, graphical and interaction features.

[2] Our context of use conception encompasses five components: the environment, the user, the platform, the particular shared knowledge and the task at hand.

to mold their own UI 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 [12], offering a great flexibility. We propose to adapt plasticity tools to novel groupware scenarios. Another handicap of CSCW is the lack of evidence in the use of complex social dynamics where the group activity takes place. In this line, groupware designers have included aspects related to *awareness*, which has become a cornerstone in computer systems design in several ways. Awareness reduces the meta-communicative effort needed to collaborate across physical distances in CSCW environments [8] and promotes real collaboration among group members. The shared knowledge should be appropriately captured, represented, integrated and promoted. However, awareness support is not systematic and developers must build it from scratch every time.

Groupware systems must be highly adaptable to new changing conditions involving not only constraints related to mobility as described above, but also to their distributed shared knowledge. In this paper we deal with the dynamic adaptation at runtime. It is necessary to implement some mechanisms to obtain a twofold benefit: (1) reaction in a proactive manner to contextual changes and (2) shared-knowledge awareness, contributing to make collaborative work successful. The infrastructure presented in this paper reuses some existing tools addressed to provide plasticity in order to integrate awareness information and exploit it as an integral part of the plasticity process.

This paper is structured as follows. Section 2 discusses some related work. Keeping in mind that our goal is to integrate awareness mechanisms in a certain infrastructure of plasticity, in section 3 we describe the approach and infrastructure of plasticity we consider the most appropriate. Section 4 presents a description of the software architecture we propose and reports some guidelines of abstraction towards a major flexibility to compose the generic *application framework* [3] we pursue. Finally, some conclusions and further work are explained.

2 Related Work

Many authors have proposed different elements necessary for obtaining a real collaboration. We present some related works to support collaborative systems in which proactive aspects and/or awareness mechanisms are treated.

Collaborative environments supporting proactive adaptation. It is worthy pointing up the research on collaborative work based on the Collaborative Filtering approach. In this line, some authors have emphasized their research on adaptive systems (proactive adaptation) based on the group member's interests. In particular, Barra [1] describes an adaptive system for group navigation on the web whose goal is to provide collaborative and adaptive navigation to users groups sharing a "common interest" on the web. However, the most part of these

[3] A semi-complete application that can be customized to produce particular applications.

systems is only focused on different static user aspects, providing different kinds of prefixed profiles.

Collaborative environments supporting heterogeneity and proactive adaptation. One of the most relevant and complete work is the one developed by Favela et al. [4]. Their research is applied in the healthcare field. They combine interactive public displays together with handhelds, towards the development of a pervasive hospital environment. The integration of proactive components in an agent-based architecture offers information relevant to the case at hand, apart from context-aware and personalized information to the user. We can point up that their approach is totally dependent on the server, implying that some problems like network failures could have serious consequences in critical environments like hospitals. In that way, our approach intends to reduce this degree of dependence, tending towards client-server architectures that obtain an operational balance between both sides. Its idea is to make client devices autonomous to a great extent by means of low resource-consuming programming techniques that can be supported in small devices, as explained in section 4.1.

There are many works related with frameworks to dynamically support context-awareness. Marsic [7] has developed a data-centric framework for synchronous collaboration of users with heterogeneous computing platforms, allowing clients with different computing capabilities to share different subsets of data based on XML. The interface is customizable according to the context and the user needs. However, the level of shared information is quite limited, and what is more important, it is static. The shared-knowledge awareness [2] is not handled.

Collaborative environments supporting awareness and proactive adaptation. One of the most relevant works that integrate awareness mechanisms is the one proposed by Correa and Marsic [3]. They have developed an extensible and generic architecture to support awareness in heterogeneous collaborative environments under a semantic consistency approach, what is their main innovation. Their architecture not only provides awareness, but also allows the adaptation, although only related to resource constraints. They show that awareness support implies a trade-off between the degree of awareness and the network usage. In fact, our work is in the line of pursuing this balance. Another common point with our work is the development of an application framework. They plan the development of an adaptive version of their architecture to awareness issues.

As we have observed, there is a lack of guidelines about how to integrate aspects as adaptivity, context-awareness and shared-knowledge in the same tool. Our work intends to support the development of computing collaborative environments that integrate all these aspects.

3 Initial Approach and Infrastructure of Plasticity

The infrastructure chosen for our proposal is based in our dichotomic view of plasticity [9], which divides the plasticity problem in two different challenges with two clearly delimited goals that make up an extension to the Thevenin and Coutaz concept of plasticity [12]. They are called *explicit plasticity* and *implicit plasticity* [9]. We match these two issues respectively with the stages of design (construction, sometimes a reconfiguration of an existing UI) and execution (specific readjustments at runtime) that the UI has to withstand over the whole system's lifetime. To be more precise, *explicit plasticity* tackles relevant changes in the UI, caused by unforeseen situations that involve a reconfiguration of the UI (e.g. changes in the computing device). 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". *Implicit plasticity* is in charge of providing proactive adaptation (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 (e.g. changes in the daylight level or in the user's location), 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". Clearly both challenges require different modelling, strategies and tools; hence they need to be studied and developed in different frameworks. This division in two goals is what we call a "dichotomy".

Under this twofold perspective, the infrastructure of plasticity consists of combining two different engines framed in a client-server architecture. These engines are called *Explicit Plasticity Engine* (EPE henceforth) and *Implicit Plasticity Engine* (IPE henceforth), respectively. The EPE consists of an automatic tool of production of plastic UIs, as in a design stage. The IPE consists of a runtime adaptive engine with the capacity to detect the context and react in order to adapt the UI to the contextual changes on the fly, providing thus the proactive adaptation pursued on the client side. In this line, our interest is focused on developing a generic framework easily customizable to IPEs for particular systems. This is what we call *Implicit Plasticity Framework* (IPF henceforth).

This architectural framework allows delimiting clearly both goals, to be solved in both engines, which are managed in an alternative, iterative and complementary manner. The goal is to give feedback to the plasticity process without discontinuities, keeping both sides in continuous updating. Under this approach, the client only resorts to the server when he/she needs a reconfiguration of the UI -unsolvable locally by the IPE-, propagating to the server the contextual changes that require to be accommodated to the new situation. We can sum up the benefits of our infrastructure in these three ones: (1) an operational balance between both sides; (2) autonomy to perform adaptivity on the client side (that reduces dependence to the server and possible communication failures); and (3) real time reaction to certain contextual changes, contributing to a proactive (implicit) adaptation. Figure 1 shows the overview of the process described, as well as the delimitation between the two sub-concepts of plasticity. The EPE is out of the scope of this paper. A detailed description can be looked up in [10].

In order to integrate the awareness information, the key point is to include the evolving group state and the real time group constraints in the characterization of the context of use. This information is captured and represented in the

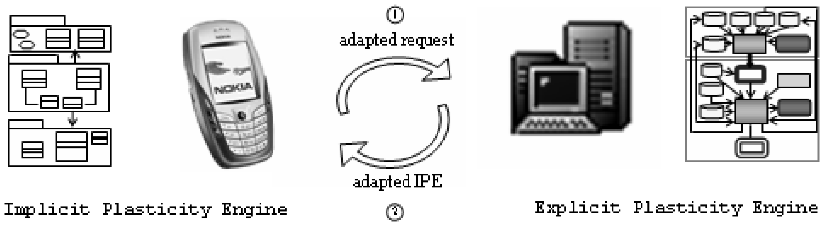

Fig. 1. Overview of the plasticity process

client-side in order to be considered at runtime. However, it is essential sharing it with the rest of the group. This is why he/she propagates it to the server, which will be able to gather and model an overall group knowledge, through the EPE.

4 Software Architecture for the Client-Side

4.1 Design Requirements and General Structure for the IPE

Taking into account that our final goal is to develop a generic framework to easily derive the suitable IPE for a particular system (the IPF), we apply the most orthogonal design strategies to obtain the most challenging design requirements. Particularly, we must guarantee certain properties such as: (1) transparency in adaptation; (2) reusability; and (3) orthogonality. In particular, orthogonality is essential in order to the adaptive mechanisms be handled independently, so that they can evolve individually, avoiding conflicts and promoting reusability. This property is especially important in systems where a lot of dimensions are presented, such as collaborative environments. In order to guarantee these properties we need to apply a separation of concern technology.

We conceive an IPE as a software architecture divided into three layers. The two first are: (1) the *logical layer*, which contains the application core functionality; (2) the *context-aware layer*, which controls and models the real time constraints (the contextual model); in the case of collaborative applications, all the information related to the communication and coordination actions that affect to the whole group state: the *shared-knowledge* [2] from the perspective of the user at hand. This information will be kept updated for further use, in order to provide awareness. Finally, the third layer is an intermediate layer responsible for applying the adaptation over the core system according to the contextual model (*context-aware layer*) following, as mentioned before, some sort of separation of concern technology. The approach chosen for this layer is the Aspect Oriented Programming [6] (AOP henceforth). The reasons that justify this decision are out of the scope of this paper. It can be looked up [11] for a detailed justification, as well as an introduction to AOP. AOP guarantees the three properties mentioned before, obtaining the maximum modularization for the adaptive mechanisms, and what is more important, without affecting the software structure of the underlying system (transparency). AOP offers these goals encapsulating the treatment of each real time constraint (e.g. the related to groupware) in separated and concentrated program units called *aspects*[4] (orthogonality). Hence we call the intermediate layer (3) *aspectual layer.*

From a general collaborative viewpoint, the *aspectual layer* has the responsibility of promoting communication events and actions focused on improving the coordination of the group activities. To do so, it is in charge of interfering implicitly the situations along the system execution in which these initiatives could be beneficial for the group, and augment there the core functionality triggering this kind of actions, as well as catching the context in order to construct the *shared-knowledge* from the particular viewpoint of each user. These actions are triggered by means of the AOP mechanisms, trying to improve collaboration, and they make up the extra-functionality that the IPE introduces without affecting the underlying system. Furthermore, due to we use a combination of metadata and *aspects* -whose explanation is out of the scope of this paper-, coupling between the base system and the adaptation mechanisms is removed (reusability). We can say that the *aspectual layer* acts as a transparent link between the other ones. Figure 2 depicts a sketch of an IPE for a generic collaborative system. We include two *aspects* to tackle the two goals mentioned: the `Communication` *aspect* and the `Coordination` *aspect.* The role of the `coreAppAnnotator` *aspect* in figure 2 is also out of the scope of this paper. The Shared-Knowledge-Model component corresponds to a representation of the *shared-knowledge* from the particular user viewpoint.

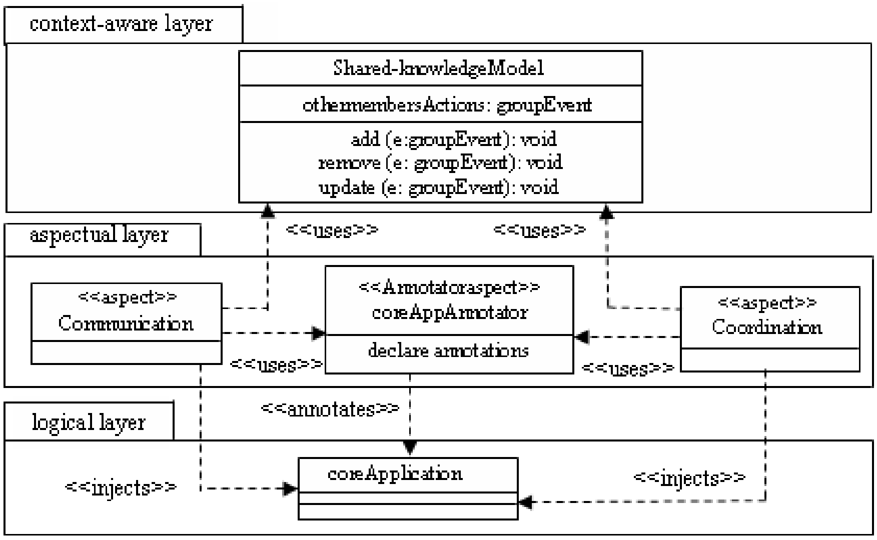

Fig. 2. IPE for a generic collaborative system

[4] Program units that interfere the core functionality injecting new code or modifying the base code.

4.2 Further Guidelines Towards Abstraction: The IPF

In the design of our IPF, according to the experience extracted from the IPEs built up to now, we take into account some considerations in order to achieve system-independence and reusability. Let us see them according to each issue pursued: different adaptation mechanisms, different contextual needs and different domains of application.

Adaptation mechanisms. In order to avoid system-dependences, and to obtain "universal" adaptation mechanisms, we resort to *aspect* hierarchy and to some refactoring steps. Thus, references to the name of certain methods, classes or particular APIs are encapsulated, redefining the associated elements in *sub-aspects* conveniently specialized. This is also the appropriate strategy if we need to define different ways to interfere the base code behaviour following the AOP mechanisms. Another strategy to achieve abstraction is refactoring in the *aspects* code (the *advices*[5]). In effect, sometimes is not necessary to define the complete advice in *sub-aspects*, but only a method that encapsulates special needs or variabilities. Then, we use advice refactoring. This idea in particular corresponds to the *Template advice idiom* [5]. We can use other types of refactoring or AOP-specific patterns to make good aspectual designs.

Application domains. We are planning to reuse our IPF for different domains deploying libraries of *aspects*. Thus, each particular application would be able to establish the set of concerns it needs to manage. For example, in an archaeological site considering the daylight constraint to adjust the UI is required. However, in an indoors museum guide this concern is useless. In a tele-aid system another kind of concerns are required to assist high-mountain rescues. It would be useful to build a package of *aspects* related to mountain conditions.

Contextual needs. In a similar way, we need to adapt the *context-aware layer* to the *aspectual* one, in order to map *aspects* with data stored in the contextual model. We need to obtain flexibility also in the *context-aware layer*. As long as this layer is based on the object-oriented programming, we use hierarchy of classes to build their components in a generic way.

5 Conclusions and Further Work

Assuming that mobile solutions can offer large-scale solutions in supporting coordinated work, it is recommendable to reuse as far as possible the work already realised to solve problems inherent to mobility in the groupware work. In this line, plasticity tools intend to offer a solution to most of the issues related to groupware. Moreover, groupware activities need to be designed providing awareness mechanisms to share the group understanding among group members. We have presented an infrastructure of plasticity in which to integrate awareness

[5] The code to be executed when aspects intercept the base code. Equivalent to methods in classes.

mechanisms, as well as how to accommodate the client-side software architecture to support collaborative activities. The aim is to combine plasticity and awareness goals, until now following separated ways, making group issues an integral part of the plasticity process.

The infrastructure of plasticity proposed follows a client-server architecture model not centered in the server as usual is presented in the literature, but adjusted to our dichotomic view of plasticity, which looks for an operational balance client-server. This approach promotes a better collaboration between devices and less constrained mobility conditions, contributing to autonomy and robustness at the client side, as well as to a real time reaction during the task performing. This approach is in the line of obtaining a trade-off between the degree of awareness and the network usage, identified by Correa and Marsic [3]. Moreover, the software architecture for the client-side is based on low resource-consuming program units (*aspects*) that support awareness mechanisms and adaptivity in compact limited appliances, scarcely increasing the size of the final code. As a result, this architecture becomes suitable for the pervasive design. Additionally, the integration of the adaptive mechanisms with the base system becomes a seamless process because of: (1) any refactoring step or modification in the software structure from the initial system is needed; and (2) coupling with the base system is also removed due to we use a combination of metadata and *aspects*, promoting so reusability.

As further work we plan to arrange and deploy a hierarchical library of generic *aspects* that might be included in the groupware design. In a similar way, we are developing the groupware facet in the server side, constructing an EPE to tackle the design stage. Additionally, we want to specialize our work in the construction of collaborative systems for ambience intelligence and pervasive computing scenarios. Finally, we are interested to develop some visualization mechanisms to provide shared knowledge awareness information to the whole group, using the same infrastructure proposed in this paper.

Acknowledgements

Work partially funded by Spanish Ministry of Science and Technology, grants TIN2004-08000-C03, Colciencias (Colombia) Project No. 4128-14-18008 and Project No. 030-2005, and finally to ParqueSoft-Colombia (DaVinci's Laboratory).

References

1. M. Barra. Distributed systems for group adaptivity on the web. *Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000), Italy*, 2000.
2. C. Collazos, L. Guerrero, J. Pino, and S. Ochoa. Introducing shared-knowledge-awareness. *IASTED Conf.: Inform. and Knowledge Sharing*, pages 13–18, 2002.
3. C. Correa and I. Marsic. A flexible architecture to support awareness in heterogeneous collabora-tive environments. *CTS'03*, pages 69–77, 2003.

4. J. Favela, M. Rodríguez, A. Preciado, and V. González. Integrating context-aware public displays into a mobile hospital information system. *IEEE transactions on information technology in Biomedicine*, 8(3):279–286, 2004.
5. S. Hanenberg and A. Schmidmeier. Idioms for building software frameworks in aspectj. *2nd ACP4IS*, 2003.
6. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.M. Loingtier, and J. J. Irwin. Aspect-oriented programming. *11th ECOOP'97*, 1241:220–242, 1997.
7. I. Marsic. A software framework for collaborative applications. *Collaborative Technology Workshop*, pages 10–11, 1999.
8. K.A. Palfreyman and T. Rodden. A protocol for user awareness on the world wide web. *Proc. of CSCW 96, Boston, MA, USA*, pages 130–139, 1996.
9. M. Sendín and J. Lorés. Plasticity in mobile devices: a dichotomic and semantic view. *Workshop on Engineering Adaptive Web*, pages 58–67, 2004.
10. M. Sendín and J. Lorés. Remote support to plastic user interfaces: a semantic view. *Selection of HCI related papers of Interaccin 2004. Springer-Verlag*, 2005.
11. M. Sendín and J. Lorés. Towards the design of a client-side framework for plastic uis using aspects. *Int. Workshop on Plastic Services for Mobile Devices*, 2005.
12. D. Thevenin and J. Coutaz. Plasticity of user interfaces: Framework and research agenda. *Proc. of Interact 99, Edinburgh*, pages 110–117, 1999.