Supportive User Interfaces for MOCOCO (Mobile, Contextualized and Collaborative) Applications

Bertrand David, René Chalon, and Florent Delomier

Université de Lyon, CNRS,
Ecole Centrale de Lyon, LIRIS, UMR5205,
36 avenue Guy de Collongue, F-69134 Ecully Cedex, France
{Bertrand.David,Rene.Chalon,Florent.Delomier}@ec-lyon.fr

Abstract. Enhancing interaction with supplementary Supportive User Interfaces: Meta-UIs, Mega-UIs, Extra-UIs, Supra-UIs, etc. is a relatively new challenge for HCI. In this paper, we describe our view of supportive user Interfaces for AmI applications taking into account Mobility, Collaboration and Contextualization. We describe proposed formalisms and their working conditions: initially created for designers in the design stage; we consider that they can now also be used by final-users for dynamic adjustment of working conditions.

Keywords: Interactive and collaborative model architectures, formalisms, Ambient Intelligence, pervasive and ubiquitous computing, tangible UI.

1 Introduction

The Supportive User Interface (SUI) issue is a relatively new CHI research field, as illustrated by the first workshop devoted to this issue in 2011 [8]. However, several approaches to this issue were proposed already a long time ago without calling them Supportive UIs, just as Mr. Jourdan wrote prose, … We have a relatively long experience in this approach, as since 1994 we have proposed a formalism the aim of which was to support designers' and final-users' manipulations to allow dynamic changes during execution of interactive applications, first purely interactive, then collaborative, and now mobile, collaborative and contextualized.

2 State of the Art

In the workshop mentioned above [8] several approaches were recalled such as Meta-UIs, Mega-UIs, Extra-UIs, Supra-UIs, and others presented as SEUI (Self Explanatory UI), ISATINE and its SUI as well as supportive UIs derived from Collaborative User Interfaces.

3 Our Proposals

To present our general approach and detailed contributions, we start with a brief reminder of the historical evolution of UI design. The first step was software architecture geared

M. Kurosu (Ed.): Human-Computer Interaction, Part V, HCII 2013, LNCS 8008, pp. 29–38, 2013.

towards the need to separate Presentation, Control and Application behavior. Architecture models accounted for this need first by hierarchical models such as the SEEHEIM model or ARCH model. Then a reactive agent model emerged with mainly a PAC model. In a MDA (Model-driven architecture) approach, several models and formalisms such as ours (see hereafter) were proposed. The CAMELEON reference framework [1] proposed three layered approaches with CUI, SCUI and FUI. We based our work in relation to this state of the art.

3.1 AMF Architecture Principles Model

Our first contribution for SUIs (supportive user interfaces) is in relation to our proposal of the AMF model [10]. The AMF model is structured on the basis of agents, in a way that can be compared to the PAC model [3]. The main difference between PAC and AMF lies in the generalization of the number of facets: 3 in PAC, multiple in AMF. This choice was justified by the need to acquire a clear, clean expression of interaction control. This means that our goal was to express in the control facet only different relationships between other facets, the aim of which is to express other aspects of behavior. If a new behavior such as explanation, trace collection, and so on, occurs, and if it is considered as interesting and repeatable in other applications, a new facet is created. In this way, control can be and is expressed by a graphic representation relating different facet inputs and outputs by clearly expressed administrators (Fig.1). The main goal of this graphic expression of control is to allow the designer to model this behavior in a coherent and comprehensible manner.

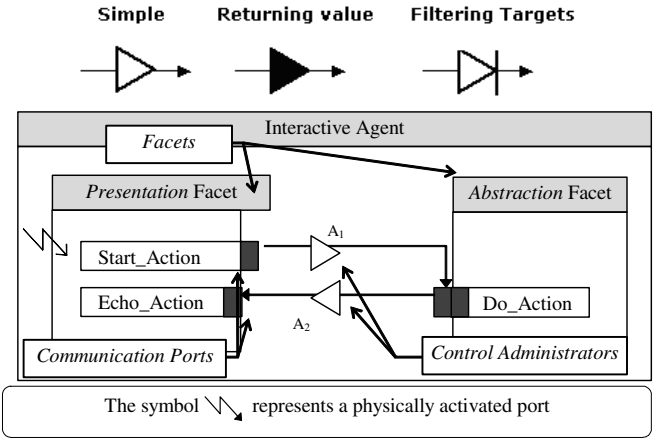

Fig. 1. AMF administrators and their use in control modeling [9]

An interesting consequence of this choice is the possibility of showing this model to the final-user and allowing him to modify the control behavior of his application by using a contextual editor. In this way, the user observing the model can modify it in order to change the behavior of his application. If he considers that the Echo_Action is not meaningful for the user, he can remove the relationship between Do_Action and Echo_Action and the corresponding Control Administrator. Consequently, the

manipulation of AMF graphic representations is a Supportive User Interface that can be assimilated to Meta-UI or Mega UI.

3.2 AMF-C Extensions

In the continuity with the AMF model approach, in 1999 the AMF-C model [10] was proposed as an extension of the AMF model for cooperative applications. In this context, the same idea of clean, clear control, expressed graphically, is used. In a distributed collaborative environment, we proposed either to replicate each AMF agent on each workstation, or to fragment each AMF agent and locate only the presentation facet on each workstation. One of the possible implementations is the "replicated mode". In this case, each AMF agent is replicated on each user workstation, and a synchronization mechanism is used at network level to propagate manipulation echoes to other collaborating actors. In this way, with an augmented set of administrators it is possible, just as for AMF, to design the control graphically. In this case, the control expresses not only individual interactions, but also, and in particular, collaborative behavior.

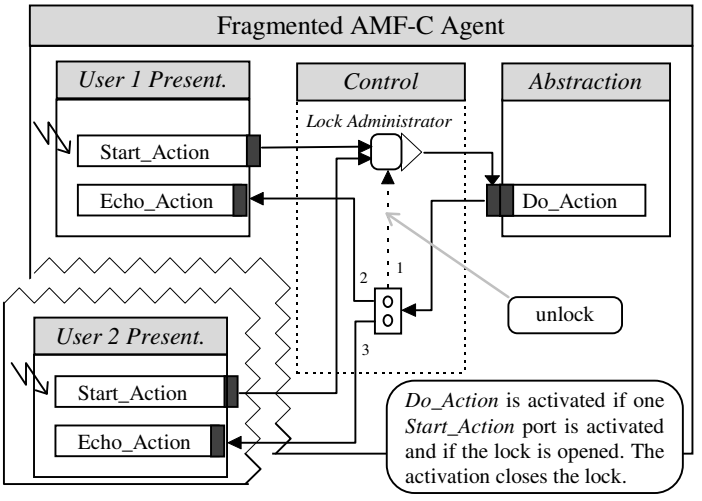

Fig. 2. AMF-C control modeling, expressing awareness.

As regards AMF, our view is that this visual programming is both appropriate and easy to use by the final-user for modifying several behavioral aspects of a collaborative system or an application using a contextual editor. We can mention, as an example of dynamic adjustment of an application behavior, the decision of awareness of a set of manipulations. This means that echoes of operations and corresponding manipulations or results are not propagated to other contributors, as shown in Fig. 2. All adjustments relating to control can be formulated in this way and at the time of execution using a contextual editor.

3.3 IRVO Model

Another model and formalism we proposed, called IRVO (Interaction with Real and Virtual Objects) is devoted to organizing interactions in augmented reality [2]. This approach is based on the idea of using a graphic expression to model the design of this kind of application. However, just as for AMF and AMF-C, we propose using IRVO as a formalism that can be given to the final-user to allow him to modify the composition of an interaction choice, i.e. create a new interaction configuration based on available real and virtual objects. In this way, several augmented reality configurations can be studied (Fig. 3). In this figure and next we present different modeling related to Lea(r)nIt a serious game oriented to Lean Manufacturing mastering which is working in real augmented environment [7]. Le(a)rnIT is an 8-player game during which each student plays an operator role in an industrial production line to understand the complexity of its dynamicity and how to improve it. Raw materials and processed materials are moved between the player's tables by a warehouseman handling a cart. After each simulated working sequence, the teacher and students debrief their working experience in order to find improvements to apply to the production line.

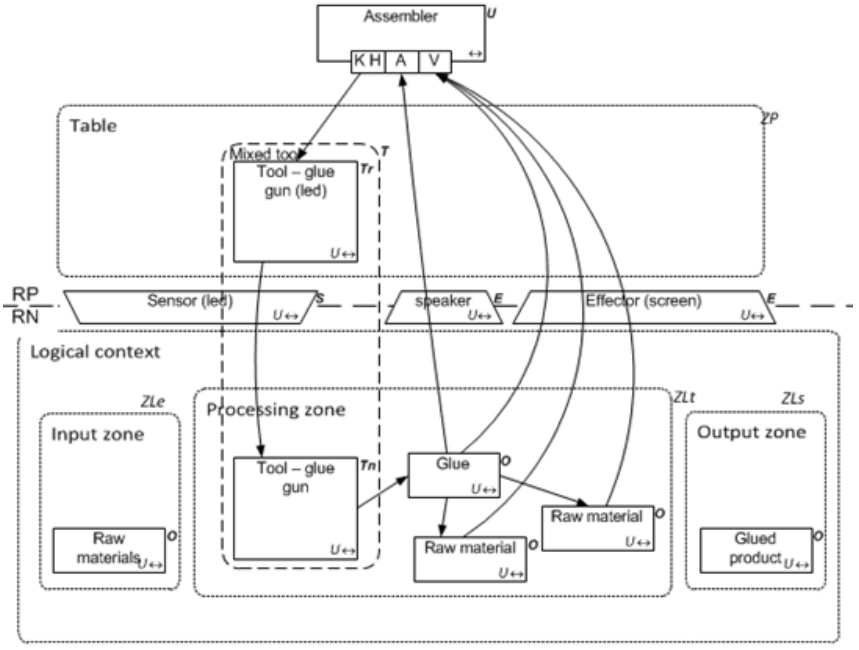

Fig. 3. IRVO modeling of a working space of Lea(r)nIt serious game

3.4 ORCHESTRA Model

ORCHESTRA [5,6] is a formalism, the aim of which is to express orchestration of collaborative applications. This formalism works at a symbolic level, and is able to

express in a music-like form, the behavior of a collaborative application between its contributors, their tasks structured in a workflow (states and transitions), and the relevant manipulated artifacts (objects and tools) in a given context. Small working periods can be organized appropriately to express workflow, with repetitions, optional sections, linking up between periods, and so on. This graphic description is decorated by working choices related to cooperative styles, coordination styles, etc.

Different key signatures express collaboration properties such as synchronous or asynchronous collaborations, collaboration modes, and styles of coordination (computational 💻 or social 😐, implicit ● or explicit ■■):

@ - **Asynchronous** with infinite answer delay
@@ - Asynchronous with limited answer delay corresponding to "**on call**" participation
& - Synchronous "**in-meeting**" cooperation
&& - Synchronous "**in-depth**" cooperation

In synchronous collaboration, two different participations must be distinguished:

- **instantaneous**, short-term collaboration, also known as implicit and expressed by ● i.e. vote activity,
- **long-term** participation, long-term collaboration, also known as explicit and expressed by ■■ i.e. sketching activity.

In short-term activity, as a vote, an implicit collaboration is appropriate (short exclusive access to the shared space), while in a long-term activity, such as sketching, explicit participation must be requested and authorised (long-term access to the shared space) either by **social coordination** (😐), i.e. one of the human contributors is in charge of this coordination, or a **computational** (💻) contributor i.e. the computer fulfils it. We graphically formulate instantaneous collaboration by a **dot** over concerned chords, while for long-term collaboration we use a horizontal line ▬ and a symbol expressing social or computational coordination (😐, 💻) i.e. coordination performed by one of the contributors or by interaction (asking for, receiving and returning exclusive access right to shared space).

An important notion in CSCW is **awareness**. Its goal is to allow different contributors to know (or not) what has been done by another contributor. It is important to decide statically (by the designer) or dynamically by the contributor himself the scope of information propagation to other contributors. Statically, we propose expressing awareness in ORCHESTRA formalism. Special marks are proposed:

- for no awareness,
- for partial awareness (for specific contributors),
- for overall awareness (for all contributors).

The main goal of this description is to allow in a comprehensible manner the design of collaborative application behavior. This description or this model is initially devoted to the designer; however, we consider that it can be used also to support dynamic remodeling of application behavior, which can be proposed to users, or at least to experienced users. The following example (Fig. 4) shows how to change the behavior

of a part of a collaborative application. This specification can be modified either at the "cords" description level, i.e. to add a new activity or tool to be used, or at decoration level, i.e. allowing change in awareness level.

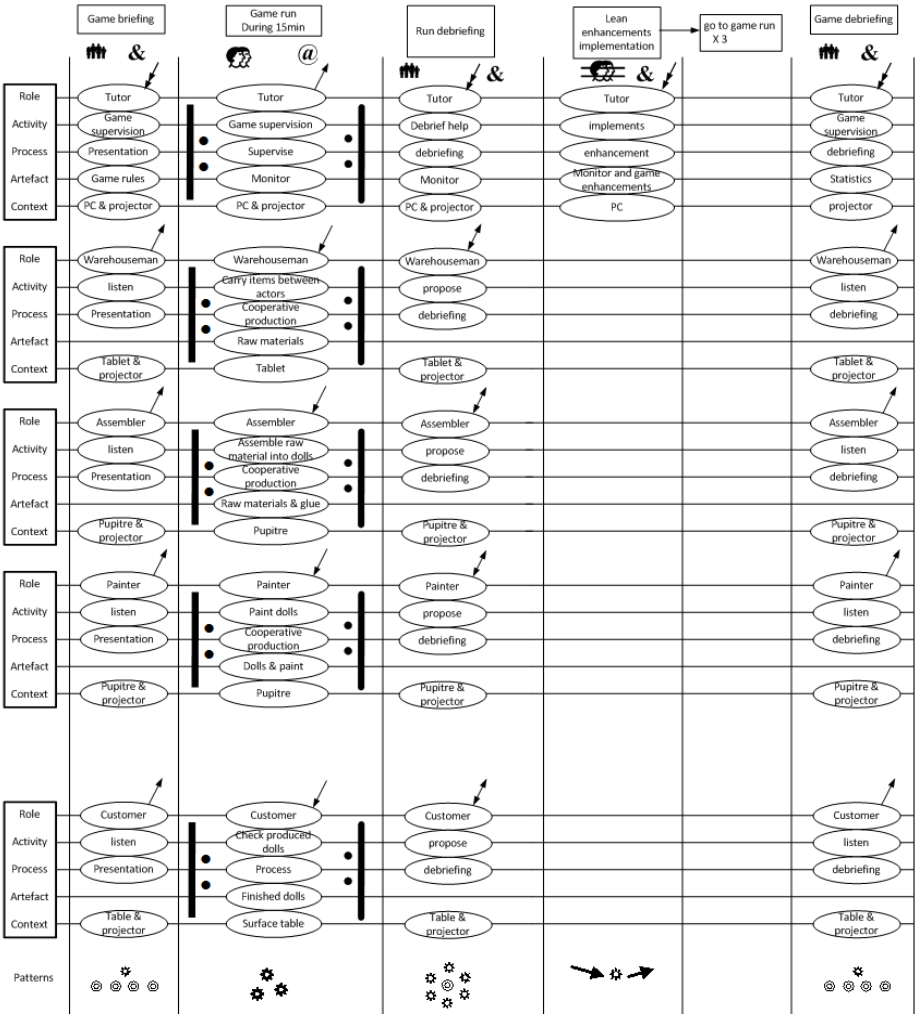

Fig. 4. ORCHESTRA modeling of Lea(r)nIt serious game

3.5 ORCHESTRA+ Approach

The ORCHESTRA mechanism is symbolic-level oriented; we thus propose to take into account progressively more precise considerations related to the user interface finalization process. We use an example to explain thus. In collaboration on interactive mono or multi-touch tables, and mono or multi-users, it is important to be able to express and manage distribution of users and actions. For this reason we are currently

working on an extension of ORCHESTRA formalism, called ORCHESTRA+, the goal of which is to progress in concretization from the conceptual user interface (CUI) to the final user interface (FUI) in the CAMELEON reference framework [1]. We can explain this using an example. For artifact manipulation, it appears important to create logical zones in which these artifacts (objects and tools) "can be alive", i.e. used, stored, manipulated, etc. This first structuring is more precise than the symbolic description of ORCHESTRA, but not enough to be considered as final, and to be used in CUI or FUI.

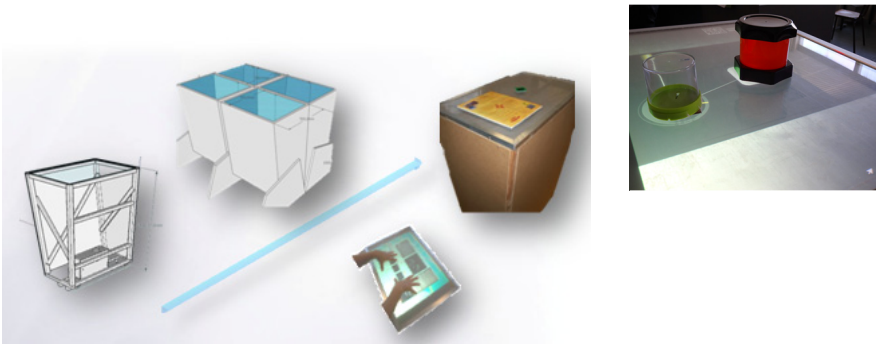

Fig. 5. Tables and tablets as collaboration support and tangible interfaces

To continue this process, we propose two complementary notions and mechanisms: **layers** and **physical zones**. The notion of layer is used to facilitate the expression of access to the artifacts. For a user or a category of users, it is possible to decide that a particular layer is either not visible, or only visible or updatable. The layer appears to be an appropriate mechanism for this access problem. The notion of physical zone is important to allow physical distribution of users' workspaces. In this way, we can dynamically determine the distribution and sharing of physical workspaces. Let us now describe an interesting situation. In a collaborative application, which is a serious game, we use several tablets and tables to work. In this context, we found it very useful to be able to dynamically determine the working conditions for each contributor.

Fig. 6. Work on different workspaces with tangible interfaces

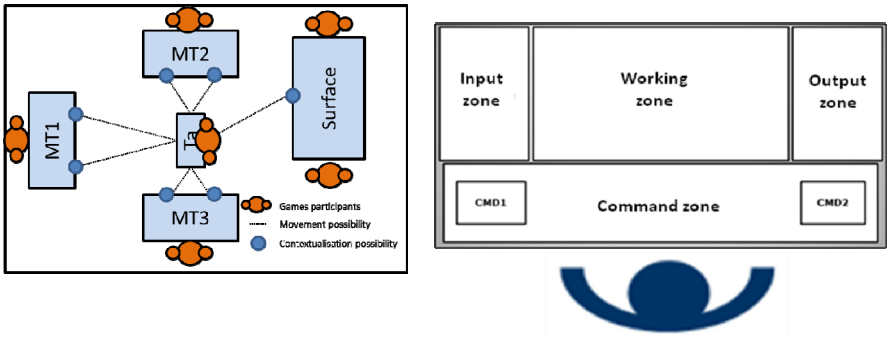

Fig. 7. a) A possible flow of materials between different workspaces (manipulation tables, distribution tablet and Surface based shop), b) A workspace with organization of different zones

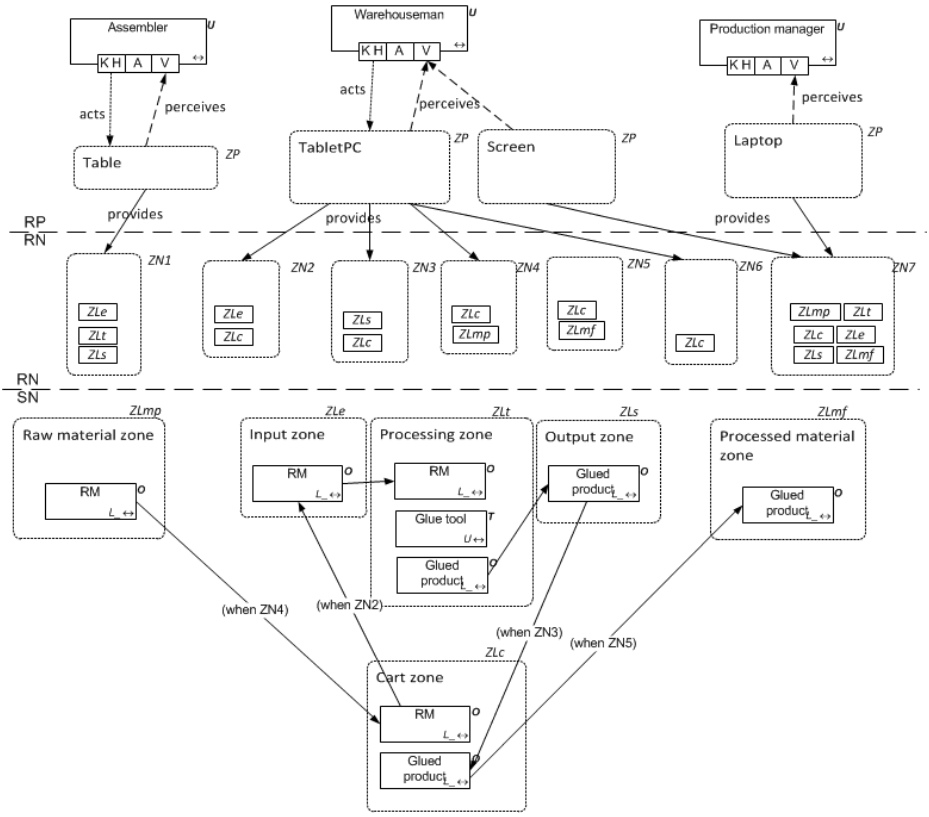

Fig. 8. IRVO Physical, numerical and logical locations modeling of Lea(r)nIt serious game

We can allow each user one tablet or table (Fig. 5), or combine all the tables to create a larger one. Moreover, we can associate different logical working zones with physical zones and thus give users the possibility to act on a zone of a table and also

on another zone of another table. We can thus create or adjust interesting working conditions (Fig. 6,7a,b). As we indicated, we can dynamically modify this distribution. We therefore created an appropriate graphic representation, suitable for the designer's work, but also and, in particular, we propose it to the final user.

The user could thus change working conditions dynamically, and allow users to work together in a table consisting of all tables, or to distribute tables and working zones according to the goal of a future working phase (Fig. 8).

4 Conclusion

In this paper we explained our view of Supportive User Interfaces (SUI). Our choice was to propose a variety of formalisms for agent-based user interface modeling (AMF), for cooperative application modeling (AMF-C), for augmented reality interaction (IRVO), and for cooperative application orchestration (ORCHESTRA). All these formalisms are initially design – developer oriented, and completed by tools and editors. As these formalisms are easy to understand, we decided to propose them also to final users, at least experienced final users, who can use a contextual editor to modify the proposed description and thus dynamically change the working application. Through their use during execution of applications, they become supportive user interfaces.

In the last case (ORCHESTRA+), we integrated 3 formalisms to describe (design) a collaborative contextual mobile augmented reality-based serious game called Lea(r)nIt. We used ORCHESTRA to describe the game and its workflow, IRVO to define augmented reality artifacts (tools and objects) working on tables or tablets, and ORCHESTRA+ to distribute work zones on workspaces of different contributors. SUI thus proposed allows us to modify dynamically the workflow of the game and to modify artifacts (tools and objects) by deciding their real or virtual being. It is also possible to remodel topological and geometrical distribution of working zones on different tables, as well as the distribution of artifacts (tool end object).

The least but not the last problem to be taken into account relates to orchestration of interactive applications in the context of their integration in a more important system. In this situation, it is important to be able not only to orchestrate these applications from their layout point of view but also from their functional rules point of view, i.e. which operation each user can execute [4].

References

1. Calvary, G., Coutaz, J., Thevenin, D.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computer 15(3), 289–308 (2003)
2. Chalon, R., David, B.T.: IRVO: an Interaction Model for designing Collaborative Mixed Reality Systems. In: HCI International 2005, Las Vegas, USA, July 22-27 (2005)
3. Coutaz, J.: PAC, on Object Oriented Model for Dialog Design. In: Interact 1987 (1987)

4. David, B., Chalon, R.: Orchestration Modeling of Interactive Systems. In: Jacko, J.A. (ed.) HCI International 2009, Part I. LNCS, vol. 5610, pp. 796–805. Springer, Heidelberg (2009)
5. David, B., Chalon, R., Delotte, O., Masserey, G., Imbert, M.: ORCHESTRA: formalism to express mobile cooperative applications. In: Dimitriadis, Y.A., Zigurs, I., Gómez-Sánchez, E. (eds.) CRIWG 2006. LNCS, vol. 4154, pp. 163–178. Springer, Heidelberg (2006) 978-3-540-39591-1 ISSN 0302-9743
6. David, B., Chalon, R., Delotte, O., Masserey, G.: ORCHESTRA: formalism to express static and dynamic model of mobile collaborative activities and associated patterns. In: Jacko, J. (ed.) Human-Computer Interaction, Part I, HCII 2007. LNCS, vol. 4550, pp. 1082–1091. Springer, Heidelberg (2007)
7. Delomier, F., David, B., Benazeth, C., Chalon, R.: Situated and collocated Learning Games. In: Conference EC-GBL 2012, European Conference of Game Based Learning, Cork, Ireland (September 2012)
8. Proceedings of the 1er International Workshop on Supportive User Interfaces : SUI 2011 at the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Pisa, Italy (June 13, 2011)
9. Tarpin-Bernard, F., David, B.T.: AMF a new design pattern for complex interactive software? In: International HCI 1997, Design of Computing Systems, San Francisco, August 24-29, vol. 21 B, pp. 351–354. Elsevier (1997) ISBN 0444 82183X
10. Tarpin-Bernard, F., Samaan, K., David, B.: Achieving Usability of Adaptable Software: The AMF-based Approach. In: Seffah, A., Vanderdonckt, J., Desmarais, M. (eds.) Human-Centered Software Engineering: Software Engineering Models, Patterns and Architectures for HCI. Springer HCI Series, pp. 237–254 (2008)
11. Tarpin-Bernard, F., David, B.T., Primet, P.: Frameworks and Patterns for Synchronous Groupware: AMF-C Approach. In: Chatty, S., Dewan, P. (eds.) Proceedings of the IFIP Tc2/Tc13 Wg2.7/Wg13.4 Seventh Working Conference on Engineering For Human-Computer Interaction, September 14 - 18. IFIP Conference Proceedings, vol. 150, pp. 225–241. Kluwer B.V, Deventer (1998)