MANAGING INTER-ACTIVITIES IN CSCW
Supporting users emerging needs in the CooLDA platform

Gregory Bourguin, Arnaud Lewandowski
Laboratoire d'Informatique du Littoral (LIL), 50 rue Ferdinand Buisson, 62228 Calais, France

Keywords: CSCW, tailorability, dynamic integration, Activity Theory, Java.

Abstract: The CSCW research domain still tries to find a better way for supporting the users needs. Some groupware systems propose global and integrated environments supporting collaborative activities, but empirical studies show that these environments usually omit to support some dimensions of the work. On the other hand, some groups work with diverse applications that do not know each other. Mainly inspired by results coming from Social and Human Sciences, we believe that the complete CSCW environment cannot be defined *a priori*. However, we also believe that a global CSCW environment is really valuable for users. Taking our foundations in the Activity Theory, we aim at creating a global but tailorable environment that supports the dynamic integration of external applications and manages the links between them, i.e. it manages the inter-activities. This work is concretized in the CooLDA platform.

1 INTRODUCTION

Following the impressive rise of communication means, the CSCW (Computer Supported Cooperative Work) holds today a strong place in computer science. Many communities use groupware tools while realizing their cooperative activities. However, even if a real need for groupware exists, a quick look at the large literature helps to understand that many questions still exist about how to realize a good CSCW application. The main question that remains after years of research is how to realize an application really fitting the users needs. As an example, we can cite a recent study about systems supporting collaborative writing on the Web (Noël & Robert, 2003). The study presents some systems that are interesting because they propose global and integrated environments for the realization of cooperative editing tasks. However, the authors underline that none of these systems totally fulfils the requirements that better fits the users needs. It happens despite of the large number of systems supporting this activity. This work concludes that it is necessary to further develop existing tools while adding required functionalities, or to create new and more complete ones. These critics are characterizing a major problem in the CSCW research domain highlighted by results coming from Human and Social sciences. For example, Activity Theory (Bedny, 1997) teaches us that human activity is reflective and that the users needs are emerging from the activity, while they are realizing it. These results lead to believe that the complete CSCW system cannot be defined *a priori*.

The solution that is usually adopted by groups is to use concurrently different groupware applications, when they need them, each one fulfilling a dimension of the collaborative activity. The matter in this situation is that these different tools do not know each other and the coherence of the collaborative environment, the global activity contextualizing their use, is only managed by humans.

Trying to solve this problem, we are working on a platform named CooLDA (Cooperative Layer supporting Distributed Activities) that provides the means to create some global and integrated but tailorable cooperative environments. In this approach, the environment is not designed to support some *a priori* users needs, but we want it to be able to dynamically integrate groupware tools supporting their emerging needs.

The first part of this paper describes how we define a tailorable global and integrated environment while characterising what we call the inter-activities approach. The second part of the paper introduces the CooLDA platform that proposes a conceptual and technical solution to these problems. The last part of the paper briefly underlines some CSCW research works that are close but different ours.

Bourguin G. and Lewandowski A. (2005).
MANAGING INTER-ACTIVITIES IN CSCW - Supporting users emerging needs in the CooLDA platform.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 134-139
DOI: 10.5220/0002529601340139

2 THE INTER-ACTIVITIES

2.1 A fine-grained integration need

As we introduced before, groups usually use concurrently different tools while realizing their cooperative activities. Each tool supports an aspect of the group's activity. From our point of view, each tool itself supports an activity existing in the context of another more general one. As an example we can consider the setting that is described in Figure 1.

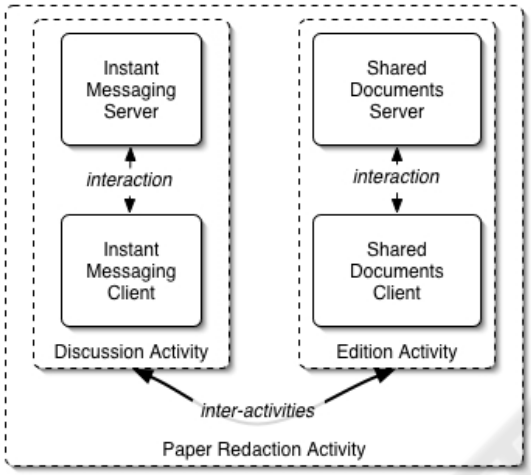

Figure 1: The inter-activities

In this situation, an instant messaging application is available over the Internet. It supports a synchronous discussion activity. In the same way, another application supports an asynchronous document sharing activity. In this example, our approach aims at providing the support for the paper writing activity that creates a context for the discussion and document sharing. We want to support the inter-activities between these two sub-activities. Such contextualization supposes that the global environment is able to start the application, control it and, in the case of a really fine integration scheme, receive some events concerning state changes in the activity it supports.

The ability to start the integrated tool seems an evidence: a user that connects to the environment and enters in the paper writing activity should have access to the instant messaging and document sharing tools. This corresponds to the minimal integration level. Such integration scheme can be found in web sites providing simple links towards different applications.

In the case of a finer integration, the role of a user in an activity should influence its roles in the linked activities. For example, in a distance learning activity like a *course*, the *professor* role may imply the *presenter* role in an activity supported by a WYSIWIS (What You See Is What I See) document sharing application and the *speaker* role in an associated audio conference. To do this, the platform that supports the course has to control the tools implied in this global activity. We can also imagine that an action that is triggered in the global activity has direct repercussions on the sub-activities, i.e. on the integrated tools. These repercussions may be different according to the different roles played by the users in the global activity. For example, after its presentation, the *professor* may decide to *start an exercise* where *students* cooperatively annotate a shared document while discussing altogether. This *professor* action in the global course activity triggers a state change in the audio-conference activity, thus allowing *students* to speak when they want, and another related state change in the document sharing application allowing *students* to annotate the shared document. Those are simple examples, but they show why it is important that the global integrated environment is able to pilot tools supporting sub-activities. If this is not the case, the teacher has to explicitly change the state of each tool he uses for its course, and this, each time he switches from a presentation to an exercise.

Finally, in the case of a maximum integration scheme, the action of a user in a sub-activity may also have repercussions on the state of the other sub-activities. As an example, we can imagine a global *debate* activity using a *vote* tool and a *forum*. Proposing a motion in the *vote* sub-activity may engender a state change in the global *debate* activity that closes the discussions in the associated *forum* sub-activity. The platform should be able to receive some events that are generated by a tool, to give them a sense in the global activity according to the role of the actor, and eventually to trigger a global action having repercussions in the linked sub-activities.

2.2 A dynamic integration need

We have talked about the mechanisms that are necessary to support the links weaved between an external tool and our environment. One can notice that these mechanisms may be found in certain full-integrated environment like complex web portals. However, it is important to remember that the main difference here is that we want to support the emerging user needs. In our approach of tailorability, the users should be able to integrate tools they need and when they need them. Thus, our platform must also propose the means for dynamically creating and evolving those links. This supposes that the environment is able to support the

dynamic integration of diverse tools as linked activities, and the dynamic (re)definition of the links created between them.

Finally, we would like to underline that in our approach, this dynamicity should be offered not only to computer scientists or system administrators, but also to domain specialists, i.e. the end-users. In fact, if a teacher finds integrable tools that are interesting for its activity, we want to let him integrate them himself, without stopping the system, i.e. during its activity. This is what the CooLDA platform is designed for.

3 THE COOLDA PLATFORM

The CooLDA platform takes benefits from our experience in the CSCW research domain (Bourguin & Derycke, 2001). This work mainly takes its foundation in the Activity Theory (AT) (Bedny, 1997), a conceptual framework that has a wide audience in CSCW (Nardi, 1996). Presenting all of our previous work in the domain with the AT is out of the scope of this paper. More details can be found in (Bourguin & Derycke, 2001). However, we will start this presentation of CooLDA by defining some concepts necessary for understanding our demonstration.

3.1 A generic model for activity

CooLDA has to support the inter-activities. For this purpose, we have defined a generic model of activity-support. This model is presented in Figure 2.

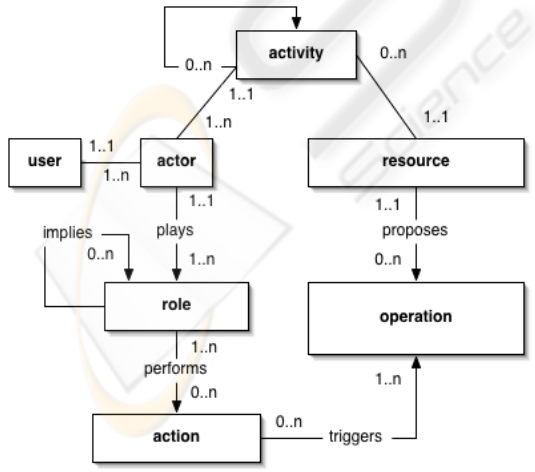

Figure 2: The activity-support model

A user is an actor in a particular activity-support. Each activity-support is linked to a resource, namely, a tool that is integrated in the platform. Each resource proposes some operations that can be triggered. An actor plays a role allowing the user to perform some actions according to the activity state. An action is realized through a chain of operations. An activity can be linked to another (sub-)activity. In this case, a role in the activity may imply another role for the user being an actor in the other activity.

With this model, we can describe how CooLDA conceptually manages the inter-activities. Going back to our example with the professor and students in a course, *professor* and *student* are the roles in the *course* activity. The course activity is linked to the *document sharing* and the *audio-conference* activities. The *professor* role implies a *presenter* role and a *speaker* role in the respective two sub-activities. The document-sharing resource proposes an *enable/disable drawing* operation that will be mapped in a corresponding action in the document sharing activity. This action will automatically be triggered for a student when the professor performs the *start exercise* action in the course activity. Several other points should be described in this scenario. It is not the purpose of this paper to describe them all here, but those we have explicated let understand how our generic model enables the specification of the inter-activities managed by CooLDA. We now would like to describe how we realize the dynamic inter-activities management from the technical viewpoint.

3.2 A simple dynamic integration environment

One particularity in our project is that we do not want to develop ourselves the tools that may be integrated in the environment and we would like to let users integrate themselves tools available over the Internet. The generic properties needed for integration that we have already enunciated in part 2 exist in diverse distributed environments like CORBA. The same type of elements can be found in the more recent Web Services technology (Kleijnen & Raju, 2003). These elements support the dynamic discovery and invocation of objects (or services) and methods over a network.

Another particularity of our project is that we are actually working with teachers involved in the development of distance learning situations. These teachers use concurrently different tools supporting their distributed collaborative teaching. They are interested in using CooLDA as a global integrated environment supporting scenarios related to their teaching activities. The tools they use are mostly

simple Java applications they find over the Internet and, for some, Java applications developed by students at university..

According to these constraints, we have developed the current version of CooLDA using a very simple integration environment : the standard Java Virtual Machine (JVM). The description of CooLDA in the rest of this paper will be linked to this choice. However, the principles we are going to describe are transposable in other technologies.

3.3 Supporting inter-activities

Java enables to download and dynamically create instances of any Java application whose classes have been made available over the Internet in a simple Web server. An URL allows downloading the client part of a groupware application. Thus, the CooLDA client is itself a Java application which instantiates client tools in its own virtual machine.

As the instance of the object-application is created inside CooLDA, we benefit from the Java message sending mechanisms for supporting the interactions between activities. These mechanisms are synthesized in Figure 3.

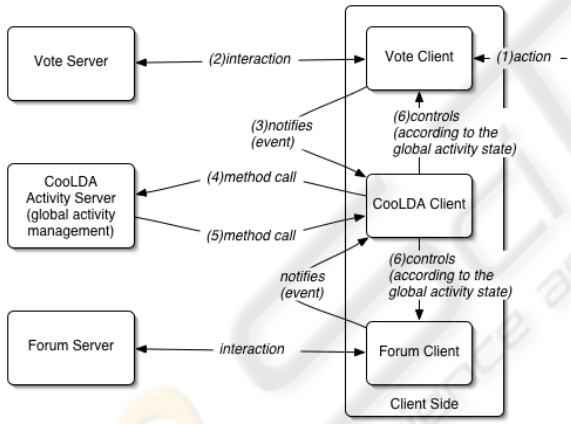

Figure 3: A debate inter-activities scenario.

This figure takes back to the debate example. The action (1) of a user in its vote client by proposing a motion warns (2) the vote server. This corresponds to the "normal/external" functioning of this distributed application. The CooLDA client, which is in the same JVM, can receive a message or event (3) that is generated by the vote client. This message is thrown (4) to the CooLDA server (actually using CORBA) that determines the sense of this local action in the global debate activity. This may imply a call-back (5) on the CooLDA clients that will then control the linked sub-activities, i.e. it will cause needed method calls on the other client applications. In this example, the CooLDA client will ask to the Forum client to change its activity (sending a message to its server) towards a state where discussions are closed.

It is important to notice that each separate application is free to choose its communication protocol (CORBA, SOAP, RMI, JSDT, JXTA, etc.) to manage its own activity. On the other hand and for a fine integration, it is necessary that the application designer has defined methods on the client side enabling the communication with a tierce environment. In our example, the Forum client should propose a method like *stopDiscussion()* that changes the Forum activity state managed by the Forum server. This server will itself warn the other Forum clients interested in this Forum activity. Any integration approach needs the creation of such interface. However, our approach does not constrain the application designer to implement a particular extra communication protocol. Only the basic message sending mechanism, available in any JVM, is solicited. Moreover, a part (if not all) of this needed interface certainly already exists in the client. It is indeed probable that the application designer has created a menu or a button calling the *stopDiscussion()* method. In fact, such a method belongs to the Forum activity abstraction level.

3.4 Inter activities dynamic specification

3.4.1 Using introspection

We want to let users themselves weave the links between their activities. A particular link will for example define which methods of the integrated tool will be called by CooLDA to create a configuration according to the role of a user in a global activity. In the same way and as in the debate example, a state change in the global activity will use these links to be transmitted to the associated activities.

For this dynamic (re)definition of the links, we use the Java reflective mechanisms, particularly the introspection. When an application-object has been downloaded from a Web site, it is possible to dynamically get a list of its methods. These mechanisms are similar to those used in Sun's BeanBox that proposes to graphically link JavaBeans. The main difference with CooLDA is that we use them to create links between high abstraction level components like a chat tool.

Another difference is that we want to help end users to integrate themselves the tools they need in the environment. The matter with introspection is that it presents many methods that should not be used for specifying integration. Moreover, the name

of the methods and the parameters types are close to the underlying language abstraction level, and then, the description is difficult to read and understand for a non-programmer. This is why, we need a technique to filter and better describe the elements presented to the user integrating a tool component.

3.4.2 The Activity Descriptor

Filtering can be realized by different means and some examples exist in classical software design. For example, the JavaBeans recommendation uses prefixes for filtering the methods related to properties. Another technique is to create a specific separated interface description. This is the role of IDL with CORBA and WSDL in the Web Services technology. However, classical interface description languages use semantics designed to inform only programmers. Java proposes another interesting mean with the BeanInfo technique. A BeanInfo allows the component programmer to create a specific interface for integration by filtering the methods and, if needed, by proposing a high abstraction level user interface that is instantiated to manipulate the component at integration time. This technique is interesting but it requires over-development.

Inspired by all these techniques but still thinking to our end users, we have decided to create a tool, named *Activity Descriptor* that helps to create an XML description of the activity supported by a tool component. The XML file contains a separated higher abstraction level description. When a user needs to integrate a tool in CooLDA, the platform uses Java introspection coupled with the tool's XML description. If the description is not found, only the standard introspection mechanism is used.

The Activity Descriptor also uses introspection. This enables to document a component whose source code is not available. Activity Descriptor presents all of the component's constructors and methods. Selecting an element uses its structure to propose a form for creating a description. For example, selecting the *ChatApp(...)* constructor in a Chat application allows to create an operation named *Start the chat* with its particular description according to the required parameters. This description will be used in CooLDA to help a user to define an action while hiding the implementation level : the user will be able to specify a specific role with specific actions realized by operations. Even if there is a direct mapping between the *Start the chat* operation and the *ChapApp(...)* constructor at run-time, the underlying programming language abstraction level is partially hidden. The Activity Descriptor tool still has to be improved, but the current version helps us to develop our approach.

We have described how it is possible to dynamically create specifications to start and pilot integrated tools in CooLDA. Even if this is not the maximum integration level, these functionalities are helpful : it allows a professor to perform a *start exercise* action in a *course* activity, this automatically triggering operations on the tools involved in the realisation of sub-actions performed by the sub-roles in the sub-activities.

3.4.3 Activity events

However, the last point is related to the maximum integration means, i.e. receiving events from an integrated tool. We already noticed that CooLDA could technically receive events from other components because they are running in the same JVM. The question is, *what to listen to ?* With the hierarchical approach defined in Java for creating user interfaces, it would be possible to register CooLDA for receiving events from each widget composing the tool. The matter is to give a sense to this event in the global activity. This is why we develop a different approach. We propose a simple framework, based on the publish/subscribe paradigm, which can be used by tools developers to extend the functionalities of their applications. This extension provides an *activity event generator* and a simple *activity event* model. Using introspection, CooLDA can dynamically discover if a tool proposes an *activity event generator* and listen to receive activity events. With the same mechanisms, the Activity Descriptor tool helps in creating a high abstraction level description of the events that may be generated by a tool. When describing inter-activities, a user can then know which event like *a vote has been proposed* can be generated by a particular tool, and specify which actions should automatically be triggered in the global activity. The main drawback of this approach is that it requires that the integrable tool use our simple framework for activity event generation. Then, this type of integration is constraining for the tool developer : the component has to be adapted for CooLDA. This seems to be the prise for a maximum integration scheme.

4 RELATED WORK

Different works have been realized by CSCW researchers using a component approach for tailorability and it would not be possible to summarize them all here. There are several differences between these propositions and ours.

Systems like in (Grundy & Hosking, 2000) only allow arranging predefined plugins components at the user interface level. We develop an approach allowing end-users to finely integrate themselves the components they need in the platform.

In (Stiemerling & Cremers, 2000) the authors develop a component model named *Flexibeans* designed for creating components that are finely and dynamically integrable by end-users in the *Evolve* platform. This approach is different because it proposes a completely new component model for composition as we try to directly use standard mechanisms. Evolve also differs from our platform because CooLDA takes directly its roots in the Activity Theory and then our view of what is a component is a little different. For example, in CooLDA, a tool that has been documented and/or adapted while its integration becomes a high abstraction level component that may be easily integrated in someone else's activity. The scenarios for inter-activities that are defined or adapted by users are also reusable CooLDA components for other users. This way and thanks to our component approach, CooLDA aims at presenting an important property inspired from the Activity Theory : the crystallization of the users experience inside the computer artefacts supporting their activities. In other words, reusing a CooLDA component is taking benefits from the experience of the actors that used and developed it during their own activities.

5 CONCLUSION

In this paper, we have shown that despite of the strong need towards CSCW, problems still exist in proposing the needed groupware applications to their potential users. We have underlined that existing global and integrated groupware environments are interesting but, as they are usually designed for fulfilling *a priori* users needs, they are not completely satisfying. Users usually solve this problem by concurrently using different groupware applications without links between them. This solution is also not satisfying because it does not provide a computer support for the global group activity that creates the context for the use of these tools. This is why we aim at creating the CooLDA platform proposing a global, integrated and tailorable CSCW environment.

CooLDA is designed to support what we call the inter-activities, i.e. the global activity giving sense to the links existing between different sub-activities. We have proposed a generic recursive model for activity that enables the specification of scenarios for inter-activities computer support. We also have identified the different properties that should be proposed by the platform for enabling the dynamic integration of tools supporting diverse activities. Finally, we have introduced an approach for end-users tailorability that proposes means for dynamic tool integration by end-users while elevating the components abstraction level. These concepts and mechanisms have been exemplified in the current version of CooLDA that has been designed for teachers involved in distance learning activities. This version uses the standard Java Virtual Machine as a simple technical integration environment.

The next step of our work will focus on usability tests and improvements. For example, the *Activity Descriptor* should evolve for creating tools descriptions closer to the end-users abstraction level. CooLDA is going to be used in real long-term distance learning activities to see how its users effectively evolve the environment. We hope that CooLDA will grow from its own use, the users experience crystallizing in the components they integrate or evolve.

REFERENCES

Bedny G., Meister D., 1997, The Russian theory of activity, Current Applications to Design and Learning,. Lawrence Erlbaum Associates, Publishers.

Bourguin G., Derycke A., Tarby J.C., Beyond the Interface: Co-evolution inside Interactive Systems - A proposal founded on Activity Theory, in Proceedings of IHM-HCI 2001, Lille, France, 10-14 September 2001, Springer Verlag, pp. 297-310.

Grundy J., Hosking J., 2000, Developing Adaptable User Interfaces for Component-Based Systems, *in Proceedings of the First Australasian User Interface Conference*, pp 17-25.

Hummes J., Kohrs A., Merialdo B., 1998, Software Components for Co-operation: a Solution for the "Get help" Problem, *in Proceeding of COOP'98 conference*, INRIA.

Kleijnen S., Raju S., 2003, An Open Web Services Architecture, *QUEUE Volume 1 , Issue 1 (March 2003)*, pp. 38-46.

Nardi, B. A., 1996, Context and consciousness : activity theory and Human-Computer Interaction., Cambridge, Ma : MIT Press.

Noël, S., Robert, J-M., 2003, How the Web is used to support collaborative writing, *Behaviour & Information Technology, 22 (4)*, pp. 245-262

Stiemerling, O., Cremers, A. B., 2000, The EVOLVE Project: Component-Based Tailorability for CSCW Applications, *AI & Society (2000), 14*, pp. 120-141.