Integrating the CSCL Activities into Virtual Campuses: Foundations of a new Infrastructure for Distributed Collective Activities

Grégory Bourguin, Alain Derycke

Integrating the CSCL Activities into Virtual Campuses: Foundations of a new Infrastructure for Distributed Collective Activities

Grégory Bourguin, Alain Derycke
Laboratoire TRIGONE, Institut CUEEP, Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq, France.
Email : { gregory.bourguin ; alain.derycke }@univ-lille1.fr

Abstract.
The integration of CSCL tools inside larger platforms like Virtual Campuses is often impossible. To overcome this problem, our general assumption is that they need a common theoretical framework for their design. In this context and turning our interest into the Activity Theory, we aim at providing new systems foundations supporting large varieties of learning modes, trying to capture the contributions of the human sciences. This has been concretised in the DARE system. DARE supports the co-constructive, expansive and experience crystallization properties of human activity. It introduces a meta-level architecture supporting Distributed Collective Activities and featuring a generic activity-support model, its meta-model, a component approach, and a distributed architecture. This provides a distributed environment where the supports for learning activities and their associated meta-activities are cohabiting, thus aiming at facilitating the development of meta-cognitive skills.

Keywords. Meta-design, integration, Activity Theory, component, distributed architecture.

Introduction

The integration of Computer Supported Collaborative Learning Activities (CSCL-A) into the broader framework of the Distributed Learning approach [16] [12] is a difficult challenge. This is not only due to the difficulty of articulating these CSCL-A with other learning practices but also to the nature of most of the CSCL tools that are mostly designed to be used as standalone. The interoperability is poor and the integration of these CSCL tools inside larger platforms like Virtual Learning Environments or Virtual Campuses is often impossible. This is partly due to the weakness of their technological openness and to their lack of clear standardised interfaces such as promoted by some standardisation bodies (see IEEE 1484 workgroups or IMS proposals). However, this technological viewpoint is only one face of the problem. At a higher level, the integration is prevented by the incompatibility between the involved conceptual models and design foundations.

CSCL tools are generally oriented towards a more or less open collaboration space, putting the learner in the centre and giving an important place to the communication processes and to the *negotiation* (a mediated co-ordination) of the flow of actions to do. At the opposite, most of the Virtual Campuses (e.g. Learning Space from Lotus) are more *process centred* because they emphasise the management of the curriculum and the *prescribed flow* of individual or collective activities assigned to the learner. Even if this analysis is probably too coarse, it seems that this apparent opposition is similar to the one existing between the Computer Supported Co-operative Work (CSCW) approach versus the Workflow Management[1] approach for supporting the collective human activities inside the organisations. The reconciliation of these two approaches is actually a hot issue in the CSCW and Workflow research domains. Because we are active in both the education and human work in the organisations application domains, we see the convergence between CSCW and Workflow Management [18] as a source of inspiration in solving the problem for integrating CSCL tools into the next Virtual Campuses generation.

Our general assumption is that we need a common theoretical framework for designing CSCL tools and Virtual Campus platforms. In the last years, we have been involved in several learning research projects where we have designed either some CSCL tools or Virtual Campuses technological platforms [9] [10] [21]. From our research activities and since the beginning, an *Activity Centred Design* [8] has emerged as a solution for overcoming the integration problem. Progressively, by integrating reflections coming from the CSCW research field we have turned our interest into the Activity Theory (AT) [13][15] to support our attempt in providing new foundations and in reframing the way we design information infrastructure supporting large varieties of learning modes. Our evolution is close to those of some others researchers in the CSCL domain. For example, Gifford and Enyedy [11] have proposed an alternative theoretical framework for CSCL that is also based on the AT. However, our work is quite different because of our involvement in the CSCW field that gives us other inputs. It also differs in the

objectives because we aim at providing new foundations for the design, trying to capture the contributions of the human sciences in order to integrate them as *first class objects* in the software design. This is a contribution for going "*toward the application of the insights of the AT to the design of CSCL environments*" [12].
An exhaustive presentation of our re-foundation in designing new infrastructures for Distributed Collective Activities is out of scope in this paper. A deeper view can be found in [3][4][5]. We will mainly focus here on our general meta-level architecture, showing how it is a solution, not only for integration, but also in providing a general framework for the design of future CSCL tools.

Our requirements for a new Infrastructure to support Distributed Collectives Activities

There are some general properties that are required for the design of these infrastructures [5]. We present here those appearing as more important and related to the AT theoretical framework. Our non-exhaustive list is inspired by recent contribution about the re-framing of the CSCW design approach like [18] and [19]. These properties can be classified following the viewpoints used for their analysis. We distinguish four viewpoints:

- The life cycle: it is important that our infrastructure supports a new kind of life cycle for the applications. This is based on the assumption that the responsibilities concerning the design of the applications will be shared between the software designers and the community of users, according to their respective roles in the organisation. This implies a *co-construction* of the computer environment. The infrastructure plays the role of an inter-mediation system between the designer providing an environment for the bootstrapping process of the attended activities, and the end-users, learners, instructors or tutors, allowed to transform this environment in respect to their goals, attitudes and skills… This is in accordance with the AT [1];
- The learning process: the computer environment has to support *reflectivity* inside activities. This is achieved using a system architecture allowing transformations of the nature of the activity support during its enactment. This property seems essential for supporting true Distributed Learning modes where learner's reflections about contents, process, tools and resources are considered as mandatory [11]. In the CSCL research field, the role of reflection over the learning process has been identified as essential [19]. This reflective nature of the human activity is also emphasized by the AT [3];
- The evolution: the CSCW community has recognised that most of the Groupwares fail due to their rigidity and their inability to support the evolution of the collective activities through continuous changes in the actors roles, choices of mediation tools, organisation of the activities… The requirement for more evolving CSCW systems seems implying a re-foundation of their design based on contributions from the human sciences, especially ethnomethodology, structuration theory and AT. Our analysis is that the property of continuous changes is in relation with the co-construction and reflectivity requirements. However, from the Human-Computer Interaction level, this can be decomposed into two properties: the *expansiveness* and the *tailorability*. Expansiveness is the possibility for the users to develop themselves the potential of their working environment, as it is developed in AT [13]. Tailorability is the activity of adapting generic computer applications to local work practices and user needs [14];
- The engineering: this represents the economical constraints and needs for open software architecture and the pressure for adopting world standards as HTTP protocols, XML, etc. The *openness* is achieved by adhering to those standard and also by choosing a software components approach [20]. This also answers to the re-use needs emerging into the fields of learning technologies accompanied by a movement towards an educational components economy [17].

The design of these new infrastructures has been concretised in the realisation of the DARE system that provides an environment for supporting co-operative activities. "DARE" signifies Distributed Activities in a Reflective Environment and has been designed to support multiple activities in an organizational context. It can be defined as a reflective-groupware, trying to fill in the "great divide" [6] between social and computer sciences by taking elements coming from these two domains for its design. We will mainly focus here on how it helps supporting the co-construction [1] and expansiveness [13] properties of human activity.

Meta-level architecture supporting Distributed Collective Activities

The general DARE architecture can be decomposed into three levels reflecting the specialisation from a generic collective activity framework to the specific applications dedicated to a particular learning

environment. This breakdown is presented in Figure 1. The three levels can be characterised by: the *foundation level*, the *composition level*, and the *user level*.

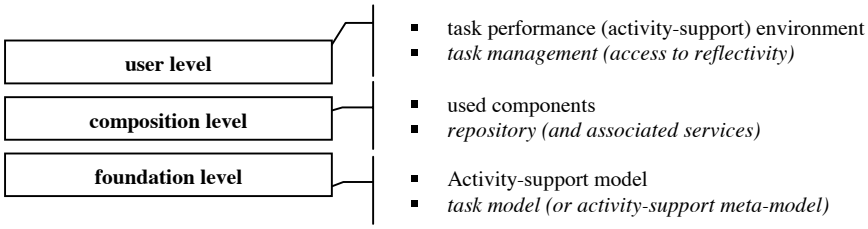

Figure 1. The three DARE levels.

Foundation level

The foundation level introduces the concepts and mechanisms that will influence the composition level and the user level. Our goal is to propose an integrative platform, creating some user environments facilitating and contextualizing the use of many different tools. More precisely, DARE aims at offering specific contexts, designed for particular purposes, and used by particular communities of users. A particular context is called an *activity-support*. Each activity-support corresponds to the specific computer support offered by DARE to the users involved in a specific activity.

An activity-support contains a set of elements corresponding to different concepts. For many reasons we have already explained in [3] and [4] we have chosen to use the concepts coming from the AT [2][15], and to support the human activity properties like expansiveness [13] and co-construction [1]. As it is presented in Figure 2, DARE introduces more concepts than those usually represented in the famous Engeström's basic structure of an activity. It is not our aim to explain here all the relations existing between the DARE concepts and the traditional AT's ones. Further explanations can be found in [5], [3] and [4].

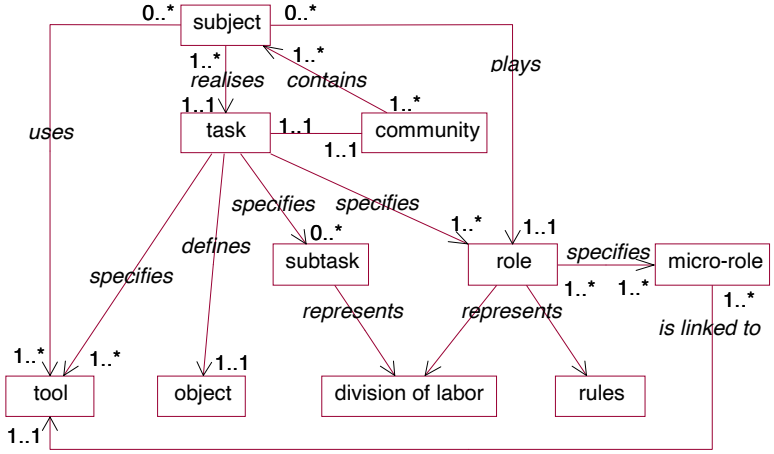

Figure 2. The DARE concepts (inspired from the Engeström's basic structure of an activity).

DARE has to support many different types of learning activities and then many different types of activity-supports can be created in the system. This is why the foundation level specifies a generic activity-support model that can be specialised for creating a particular one. Using the concepts summarized in Figure 2 we have created an object model describing an activity-support in a generic way (cf. Figure 3). An activity-support contains a set of subjects that are users involved in a corresponding activity. Each subject plays a particular role and uses particular tools including shared tools. Broadly defined, a role is composed by a set of micro-roles allowing a subject to perform some operations on the activity-support tools. For example, a *mathematics course* activity-support can be created as a specialisation of this generic model: one of the tools may be a whiteboard, the subjects involved may be a teacher and some students. The rules and the division of labour in this activity may define that the user playing a teacher role has to (and is then allowed to) put some demonstrations on the whiteboard, as the students have to look at the whiteboard but cannot write on it.

The reader can notice that this generic model is directly inspired by the DARE concepts, themselves inspired by the AT. Figure 3 shows that the specification of an activity-support like the *mathematics course* is defined as an instance of the task concept. In the same way, a role type like a *Teacher* or a

Student is an instance of the role concept, a tool type like a *Whiteboard* is an instance of the tool concept, etc.

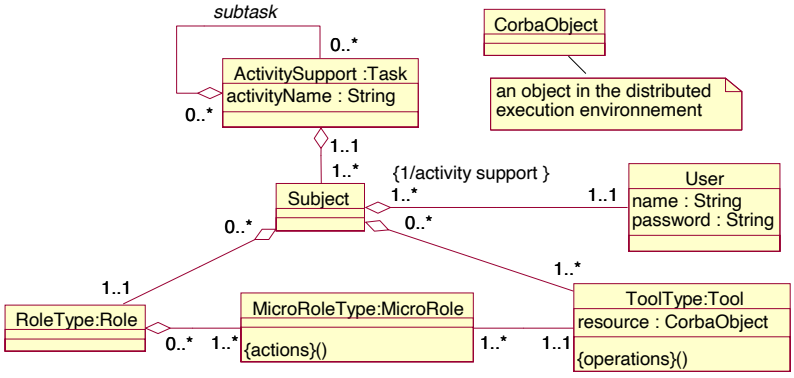

Figure 3. DARE generic model of an activity-support (UML notation).

One of our most important issues is to allow the users to co-construct or evolve their activity-support, during its execution, by rearranging or modifying the involved components (tools, roles, etc.). In traditional design, the generic activity-support model would only be known by some computer scientists, developing specific activity-supports for specific communities. From the computer scientist viewpoint, this generic model is a set of abstract classes that have to be specialised using an object-oriented language. Class, attributes, methods and inheritance concepts have to be mastered to do so. As we want to support expansiveness and co-construction of the learning environment, the question is how to make our generic model understandable and manageable from the user viewpoint? This is accomplished introducing the DARE meta-model.

A meta-model is the model of a model. Computer scientists are used to work with different meta-models. For example, UML is a meta-model that defines entities and relations for describing object models. The role of our meta-model is similar to the UML one. The difference is that the language it defines is oriented towards our domain of interest. The DARE meta-model defines a language for understanding and describing activity-supports. The description of an activity-support is an activity-support model. We have already shown that an activity-support is an instance of a task. In other words, the model of a particular activity-support is a particular task. Then, the DARE meta-model reifies what is a task, i.e. what are its components and its structure. The specification of a task contains a set of role and micro-role types, tool types and (sub)tasks. These entities and their relations are described in Figure 4.

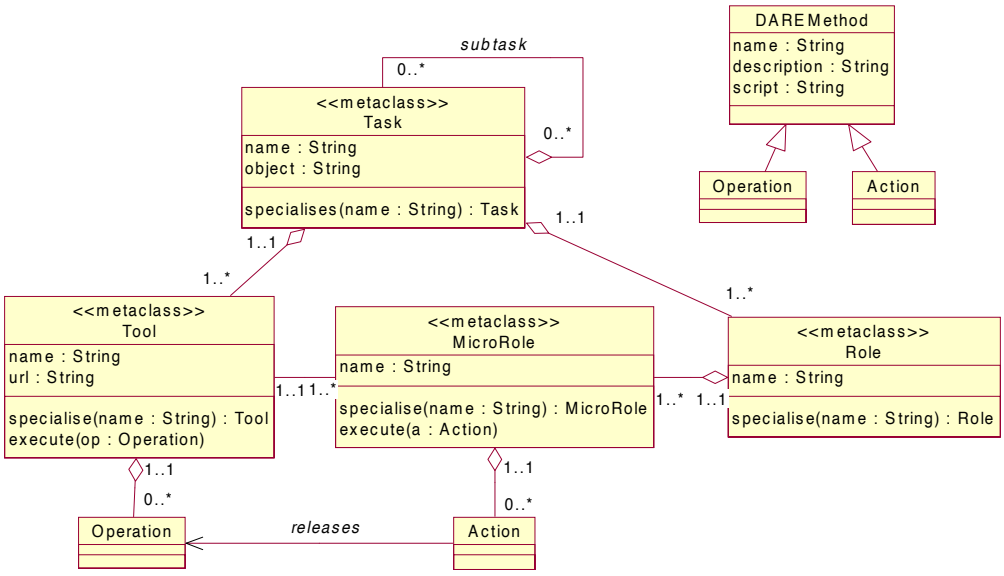

Figure 4. DARE meta-model (UML notation).

In order to support the activity-supports evolution during their own execution, a causally connected relationship exists between each task and its instances and then any modification in a task has direct repercussions on its corresponding activity-supports. For example, adding a new tool type like a chat in a task definition has the direct effect of instantiating a chat tool in each corresponding activity-support.

At a lower level, modifying the available actions defined in a particular role type has direct repercussions for the users playing this role.
Thus, the DARE foundation level uses concepts and mechanisms inspired by AT in order to create a generic activity-support model that can be specialised to create particular ones for particular communities. As we want to allow the DARE users to specialize the generic activity-support model, or to adapt a particular one to their emerging needs, these concepts are also used in the DARE meta-model. This meta-model poses the basis of a language for understanding and/or specifying tasks. This way, the modification of the elements involved in the activity-supports is not performed in terms of class, method or attribute, but in more domain dependent terms like task, tool, role, action, etc. More generally, the foundation level offers a sort of generic run-time environment allowing and facilitating the creation and management of some particular activity-supports by their own users. Of course, even if these foundations are important to support the desired properties of DARE, we know that there are more steps to be done before a user like a learner can effectively use the system. In particular, we have to create some well-defined user interfaces based on the foundations we have proposed. This problem will be developed later and we now would like to focus on the intermediate level existing between the user level and the foundation level: the composition level.

Composition level

As the foundation level defines how the elements like roles or tools can be combines and recombined together to create and evolve some particular activity-supports, the composition level more specifically addresses the elements themselves. In our approach, each element is a component and the entities described in the DARE meta-model are defining different component types.
The precedent part has shown the importance of the task concept. A task specifies and contains tool components, role components, micro-role components and subtask components. One can notice that a subtask is itself a task. Then a task is a component and any task may be or become a subtask of another. For example, consider the task corresponding to a *mathematics exercise*. The object of this task may be to calculate the surface of a disk. One of the tools used to perform this task is a calculator. Such a task is something commonly reused by teachers in the specification of a more general *mathematics course* task. The *mathematics exercise* task is typically a component that can be reused by a tierce person. It is not our aim to describe each component type defined by the DARE meta-model but one can notice that these components are DARE-specific or DARE-aware components. However, one of our main objectives is the integration of different external resources in a unified context. This is achieved through the tool component type.
An activity-support usually involves some tools. For this purpose, DARE has been built as an open system allowing the integration of external resources in the environment. These external resources are themselves *software* components like Java Beans and are often built from lower abstraction level components. Their nature can be very different. Some of them may represent generic services like group notification support, others may be groupware systems like a shared whiteboard. For us, the matter is that these components are only software components and not activity components. Their interfaces are expressed in terms of public methods, etc. Our experimentations have revealed that it is hard to understand what a software component does and can offer in an activity-support. Thus, it is hard for end users to compose them for particular needs. Our first answer to this problem only considers high abstraction level components (e.g. a whiteboard). The approach we have developed is to encapsulate these components inside DARE tool components. For example, in the actual version of DARE each tool is linked to a Java applet component. End users can perform a rough integration only by specifying the URL of the applet. The system automatically creates a link between the DARE tool component and the specified software resource. What is more interesting is that a finer integration can be done by translating the software component available methods in terms of operations that may be performed with the tool by the users. This fine encapsulation transforms a software component in an *activity-aware* component, changing the underlying paradigm from object modelling to activity-support modelling. Unfortunately, only specialists understanding both activity and software component approach concepts can achieve this. However, any DARE tool component, more or less finely defined, is ready to be used at the task abstraction level and can be brought in or removed from a task component by end-users thus evolving the specification of their activity-supports.
One more important point is that the evolution of an activity-support is realised by the users according to their emergent needs. Components are then crystallising the user's experience that has been developed during the use of the system. This way and thanks to its reflectivity, DARE offers a support for the important property of experience crystallisation inside the activity's artefacts, as it is underlined and developed in AT. It also offers the possibility to benefit from this experience by reusing these artefacts in other activities.

Following this open and reuse approach, DARE is linked to a components repository. The idea is that each component created or transformed is itself stored in the repository and available for reuse. Using DARE, the users will continually fill in the repository with their own components that are crystallising their experience. In order to help finding needed components, the repository offers some services like a criteria-search at the task abstraction level. This is only possible because the components are now activity-aware components.
Finally, it seems to be easier to compose or modify components rather than to create them from scratch. This is why DARE has also to provide some *bootstrap* components created in collaboration between DARE designers and domain specific specialists. For example, in the CSCL field that is our main domain of interest for using and testing DARE's properties, these bootstrap components may be some exercises (tasks), a set of roles (teacher, student, expert…) and tools (shared whiteboard, editor, audio conference...). These components will be used to start pre-defined activity-supports, but will be transformed by users in order to specialize them for their particular needs.

User level

A user accesses to an activity-support through a standard Web browser thanks to a particular applet called *activity applet* (cf. Figure 5). This applet offers a representation of an activity-support according to the user's role. This representation contains the set of available tools, a representation of the community (the subjects and their respective role) and mechanisms supporting some awareness properties. Each tool can be started in the user's environment from the activity applet and is automatically configured for the subject according to its role. However, we would like to underline that this applet does not correspond to the final version we will use in DARE because we are still working with HCI specialists to create more usable versions.

Figure 5. **An Activity applet.**

The user environment reposes on a distributed architecture that is depicted in Figure 6. An activity applet is connected through the network to an *activity server* holding the global representation of each activity-support. Each component involved in the users environment has a corresponding instance in the server. For example, in Figure 6, we find a user named *greg* playing the *Teacher* role. The object *:Teacher* associated to the user *greg* is an instance of the *Teacher* role in which the behaviour of a teacher is specified.

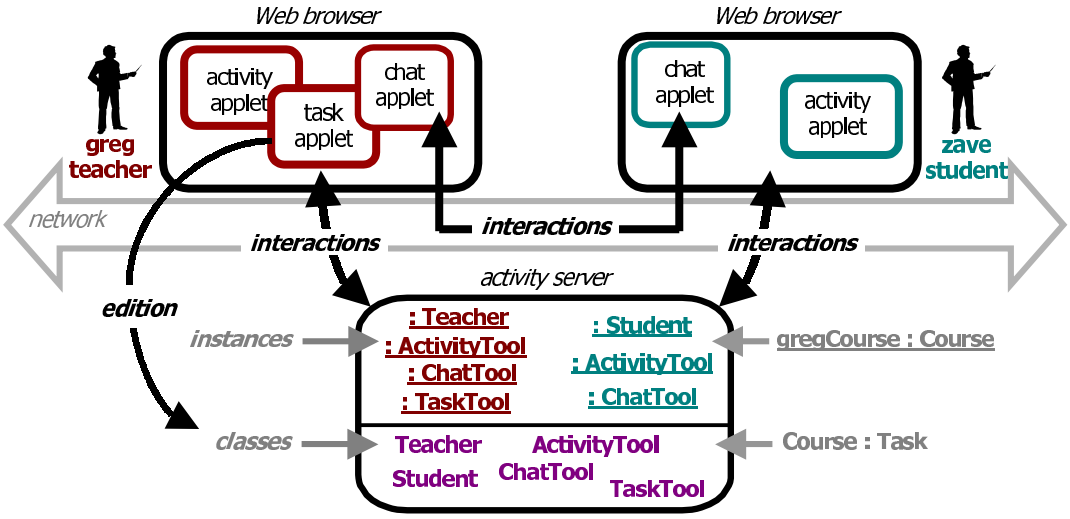

Figure 6. DARE distributed architecture.

Our reflective approach introduces new problem to be held at the user level. We have to create a user environment where the support for activities and their associated meta-activities are cohabiting. The activity

corresponds to the performance of the task. At this level, the focus is on the realisation of object. The meta-activity corresponds to another but closely linked activity where users reflect about what are the task and its elements. The activity and the meta-activity respectively correspond to the active and expansive levels of activity described by Kuutti in [13].

For supporting these properties, part of each activity-support is a meta-activity-support used to access to the task definition level. This level is reached thanks to meta-level tools. The *TaskTool* linked to the *Task applet* represented in Figure 6 is a meta-level tool allowing the users to edit the meta-level of the activity-support, i.e. the type of the involved components. DARE makes no difference between meta-level tools and the other tools. Thus, the role played by a user also affects the way he will use a meta-level tool. In our example, only the *Teacher* role allows a user to use the *TaskTool*. This explains why *greg* has started a *Task* applet in its environment, and *zave* has not.

Moreover, as the roles and tools are specified at the meta-level, they are specified thanks to meta-level tools. Here we can feel the reflective properties of any activity-support offered by DARE: the meta-level tools are used to specify their own definition and use modalities. Figure 7 represents the *Task* applet that allows modifying the tools and roles specified in the task corresponding to the activity-support where it is used. This example shows how the *Task* applet corresponding to the client side of the *TaskTool* can be used to edit itself: the user asks for editing the *TaskTool* he's actually using. One can notice that one of the other available actions for this user is the *remove* action. If the user removes the *TaskTool* from its task, its community will not be able to evolve the activity-support anymore.

Thus, it is possible to create a task that does not use meta-level tools. Such a task will be strongly constrained because the users involved in its corresponding activity-supports will not be allowed to evolve they working environment. In a learning scenario, this may be helpful, as sometimes teachers do not want their students to change the object or rules of a subtask they have to perform (e.g. an exercise)! At the opposite, introducing meta-level tools combined to negotiation tools allows a teacher to co-construct the learning environment with its students that then have the possibility to reflect on the meta-level of their activity. As we have already noticed in the introduction of this paper, this type of activity-support should be very useful in order to develop the student's self reflective skills.

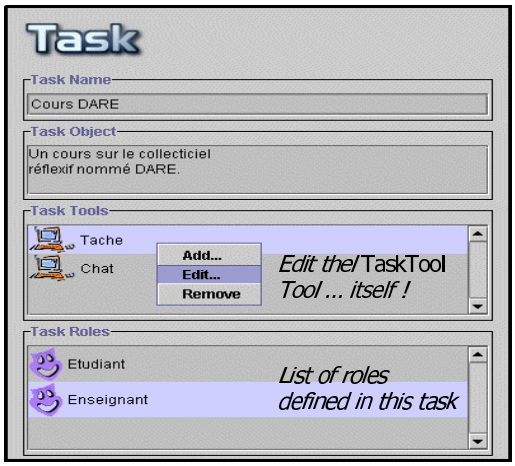

Figure 7. A Task applet.

Conclusion

Developing a new generation of CSCL systems we have taken into account the contributions from the Activity Theory as a foundation for a meta-level architecture supporting cooperative activities. This provides a "milieu" where the software components representing tasks, tools or roles are immerged and tighten together into a common theoretical framework. A particular attention has been put on the support for continuous changes realized by the users and at the run-time, thus satisfying a deeper tailorability and the expansiveness of the human activities. This is achieved by providing a reflection mechanism both at the conceptual and implementation levels. This, combined with a modern software engineering approach (not totally developed here but presented in [5]) gives a solid technical framework for developing dedicated learning systems. This development is partially performed through our participation to the DIVILAB IST European project. We are generating a CSCL system dedicated to the needs of a distributed learning environment for experimental sciences and technologies.

However a lot of work still has to be done. At the composition level, we are further developing the set of tools for the referencing and cataloguing the components. Some dedicated components more tightly coupled with the DARE environment, some kind of *DARElets* (by analogy with the servlets), will be developed to support the bootstrapping of the future co-operative activities and to ease the integration of traditional applications such as databases and Workflow Management Systems already in use educational organisations. At the user level, we want to provide more CSCL domain-specific applets based on the underlying levels, such as organisation management or even course or curriculum design-support applets. We expect that educational agents like pedagogical designers, instructors, tutors, learners, etc could use the same kind of applications with different perspectives or viewpoints. It is a way to satisfy the need for a more reflective learning process.

Notes

1. The Workflow can briefly be defined as the automated *co-ordination*, *control*, and *communication* of people and computers work in the context of *organisational processes*, through software execution into a *network* of computers. For more see WFMC [22].

References

1. Bardram J., *Designing for the dynamics of cooperative work activities*, Proceedings of the ACM CSCW'98 conference, ACM Press, 1998, pp. 89-98.
2. Bedny G., Meister D., *The Russian theory of activity, Current Applications to Design and Learning*, Lawrence Erlbaum Associates Publishers, 1997, 430 p.
3. Bourguin G., Derycke A., *A Reflective CSCL Environment with Foundations Based on the Activity Theory*, Springer Verlag proceedings of ITS'2000, Fifth International Conference on Intelligent Tutoring Systems, Montreal, CANADA, 19-23 June 2000.
4. Bourguin G., Derycke A., *Meta Groupware Design for CSCL Environments*, ED-MEDIA'2000, AACE, Montreal, CANADA, 26 June-1st July 2000.
5. Bourguin G., *Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité : le projet DARE*, Ph.D. Thesis, Informatique, n° 2753, Université des Sciences et Technologies de Lille, France, 2000.
6. Bowkers, G; Leigh Star, S. Turner, W. Gasser, L. (1997). *Social science, technical systems and cooperative work: beyond the great divide*. Lawrence Erlbaum Associates, "Computer, cognition and work" series.
7. Carstensen, P. H. Schmidt, K. *Computer Supported Cooperative Work: new challenges to systems design*. To appears in *Handbook of Human Factors,* Kenji Itoh, Tokio, 1999 (23p)
8. Derycke, A. C. "*Integration of the Learning Processes into the Web: Learning Activity centred Design and Architecture*". Webnet'98 conference, invited conference, Orlando, FL, USA, November 1998.
9. Derycke, A. Kaye, A.(1993).Participative modeling and design of collaborative learning tools in the CO-LEARN project. In G. Davis, B. Samways (eds), IFIP, Teleteaching 95 Conference, Trondheim, August 20-25,North-Holland, Amsterdam, , pp. 191-200.
10. Derycke, A. Viéville, C. (1994). Real-time multimedia conferencing system and collborative learning. *Collaboration Dialogue Technologies in distance education*, Verdejo, F., Ceri, S. (eds), NATO ASI Series, Springer Verlag, Berlin, 1994, pp. 236-256.
11. Gifford, B. R. Enyedy, N.D. *Activity centered Design: Towards a theoretical framework for CSCL*. In Proceeding the CSCL'99 conference, December 12-15, 1999, pp 189-197.
12. Grabinger. S. *REAL Strategies and Distributed Learning*. EuroConference98, Aveiro, Portugal, September, 1998. Invited conference, http://ceo.cudenver.edu/~scott_grabinger
13. Kuutti K., *The concept of activity as a basic unit of analysis for CSCW research*, Proceeding of the second ECSCW'91 conference, Kluwers Academics Publishers, 1991, pp 249-264.
14. Morch, A. Mehandjiev, N. *Tailoring as Collaboration: the Mediating Role of Multiple Representations and Applications Units*. "Computer Supported Cooperative Work" journal, vol9, 2000, Kluwers Academic Publishers, pp75-100.
15. Nardi B. A., *Context and consciousness : activity theory and Human-Computer Interaction*. Eds., Cambridge, Ma : MIT Press, 1996.
16. Pea, R. D. *Distributed Multimedia Learning Environments: why and How?*. In Interactive learning Environments, vol. 2, Issue (2), 1992, pp 73-109.
17. Roschelle, J. Kaput, J. Stroup, W. M. Kahn.T. *Scalable Intergration of Eductional Software: exploring the Promise of Component Architecture*. http://www.jiime.open.ac.uk/98/6/roschelle-01.html .
18. Schmidt, K. Simone, C. *Mind the gap! Towards a unified view of CSCW*. In proceeding of COOP2000, Sophia Antipolis, France, 23-26 May 2000, INRIA (16 p).
19. Sorensen, E. K. *Intellectual Amplification through Reflection and Didactic Change in Distributed Collaborative learning*. In Proceeding the CSCL'99 conference, December 12-15, 1999, pp 582-589, http://kn.cilt.org/cscl99 .
20. Szyperski, C. *Component Software Beyond Object-Oriented Programming*. ACM press Edition/ Addison Wesley, 1997.
21. Viéville C., Derycke A. (1998). Self-Organised Group Activities Supported by Asynchronous Structured Conversations. Proceedings of the IFIP conference on "Virtual Campus: trends for higher education, and training", Madrid, Spain, November 1997, F. Verdjo, G.Davies (Eds), Chapmann & Hall, London, 1998, pp 191-204.
22. Workflow Management Coalition, http://www.aiim.org/wfmc/ .