About the Complexity of CSCL Systems

Jacques Lonchamp,
LORIA-University of Nancy, BP 239, 54506 Vandoeuvre-lès-Nancy, France,
jloncham@loria.fr

Abstract: CSCL systems must deal with both the general complexity of supporting people doing things collaboratively through computers and the specific complexity of constructing artificial situations in which collaborative learning processes are expected to occur. This paper emphasizes three high level requirements for designing-in-the-large rich and malleable CSCL systems dealing with that multiform complexity. For each requirement the paper describes solutions taken from the Omega+ effort for providing a generic and flexible synchronous CSCL framework.

Introduction

Technology-supported collaborative learning systems are intrinsically complex. They must deal with both the general complexity of supporting people doing things collaboratively through computers and the specific complexity of constructing precise artificial situations in which collaborative learning processes are expected to occur. Simplistic tools which support a single task in a predefined situation, which are not adaptable to different conditions, which suffer from restrictive constraints for installation and use, cannot expect to be largely adopted in real learning settings. During the previous CSCL Conference, future technologies were characterized as "richer and appropriate for various collaborative settings, conditions and contexts" (Dimitracopoulou) and "reconfigurable, adaptative, offering collections of affordances and flexible forms of guidance" (Suthers). Just providing a collection of artefacts and mechanisms that have been demonstrated to be correlated with effective learning in different contexts, in a 'Swiss Army Knife' style, is not sufficient for dealing with these complexities. CSCL support requires sophisticated and powerful *integration, customization and evolution technologies*. This paper emphasizes three high level requirements for designing-in-the-large rich and malleable CSCL systems. Each section briefly introduces a requirement, mainly in the light of Activity Theory (AT), and describes possible solutions taken from the Omega+ effort for providing a generic and flexible synchronous CSCL framework (Lonchamp, 2006).

A Reflective Architecture

AT explains that the structure of any cooperative activity is dynamic and continuously evolves. Tools alter the activity and are, in turn, altered by the activity (Jonassen & Rohrer, 1999). A computerized supporting system is a mediator which should continuously reflect the current structure of the supported activity. The most obvious way to achieve that conformance is to provide a *reflective system*, i.e. a system which includes an explicit representation (model) of the activity. The behaviour of such a reflective system depends on that (continuously queried) representation and changes when the representation is modified, thanks to the causal relationship which is implemented between the activity model and the system behaviour. Modelling cooperative learning activities both for human and machine interpretation is a big challenge in this approach. In the broader e-learning field, IMS Learning Design multi-level meta-model has been criticised both for its complexity and for its incompleteness (e.g., for dealing with synchronous collaborative activities). The solution explored in Omega+ associates a separate (sub-) model for each *facet* of a collaborative learning activity: process model, interaction model, artefact meta-model, and effect model. It makes possible to build the activity representation at different levels of abstraction, adapted to the skills and needs of different categories of users: just reusing existing models, building new combinations with existing sub-models (i.e., following a very high level configuration process), defining or customizing sub-models through high-level visual languages or low-level specification languages (including programming languages).

Process model: AT highlights the importance of *plans* for guiding work (Bardram, 1997). A plan is not a rigid prescription of work to be performed but a guide that can be modified depending on context during the execution of the work. In Omega+ a synchronous process is a sequence of phases, taking place in rooms: 'simple phases', where all participants collaborate to the same task in the same room, and 'split phases', where participants are divided into parallel sub groups performing different tasks in different rooms. A plan A→B→C does not necessarily prescribe the execution of the three phases A, B, C in that precise order. ABBC, AB, AB'C (where B' is a modified version of phase B), ABCBC are other possible execution traces, while CBA and CCC for instance have a lower probability to occur. Concretely, participants playing the predefined 'Room Operator' role have two buttons for selecting the next phase to execute, either by following the plan (Next) or by selecting any other existing phase (Jump).

Interaction model: interaction protocols implement specific discourse types relevant in collaborative learning situations. In Omega+, the set of predefined generic protocols (such as 'round-robin', 'single speaker', 'moderated free floor') can be extended with application-specific protocols. These specific protocols are defined through a set of application-related roles, a set of typed messages, a set of adjacency pairs specifying how messages types are related (e.g. question-answer) and which role can speak first. Process and interaction models together implement *scripted cooperation* as defined by O'Donnell and Dansereau (1992), i.e., a set of interaction rules and phases according to which the cooperation proceeds in order to improve the effectiveness of cooperative learning.
Artefact meta-model: another important requirement for effective collaborative learning is the combination of communication with shared work artefacts (Suthers & Xu, 2002). Omega+ provides both predefined tools (shared text editor and whiteboard) and a *generic graphical modeller for graph-based hierarchical representations* that can be customized by selecting predefined artefact meta-models or by defining application-specific ones.
Effect model: effective collaborative learning requires coaching collaboration as it unfolds. The coaching process includes a data collection phase, a phase where high-level indicators are computed, a comparison of the current state of the interaction with the desired state, and a phase where remedial actions are proposed (Jermann et al., 2001). In Omega+, users can specify into 'effect models' customized visual indicators (e.g., time series, histograms) of individual and collective performance, computed from a set of predefined low level variables. Users are expected to analyze these *customized meta-cognitive tools* for devising remedial actions such as modifying plans and protocols.

Definitional, Operational, and Developmental Malleability

In AT, subjects drive evolutions for resolving *contradictions that appear during the course of the activity*. These evolutions impact the computerized support and put a strong requirement on *dynamic (run-time) malleability by end-users*. Malleability of computerized systems is considered as a difficult issue, because the more efficient mechanisms are also the more difficult to use and many users are not willing to make the efforts necessary to use them. We distinguish three kinds of malleability.
Definitional malleability: reflective systems, as defined in the previous section, provide this kind of malleability. The system can be statically (i.e., before execution) fine-tuned for various different settings, conditions and contexts, by including in its model(s) a selected choice of structural constraints.
Operational malleability: flexibility at run-time includes both dynamic model evolution and exception handling. In the case of model evolution, a change may impact only the enacting model or may also impact the template model in the model library. In Omega+ room operators can change interactively the enacting process model (e.g., change an existing phase type or add a new one). Changing template process models should be a collective decision and is discussed in the following section about meta-level support. In the case of exception handling, users with the corresponding rights can dynamically relax or sidestep a given constraint without changing the model itself. The system should be in charge of making other users aware of these rule breakings. In Omega+ predefined operations are provided for handling simple exceptions. For instance, if a learner cannot take the floor during a phase including a 'round-robin' interaction protocol, a menu item allows the room operator to skip to the next learner in the circle.
Developmental malleability: some changes, such as integrating external components that must communicate with other components through specific event types, cannot be performed without modifications at the code level. An example is discussed in (Lonchamp, 2006). In the last version of Omega+, end-users with basic programming skills, can also statically add operational semantics to graph formalisms (like Petri nets or state machines) by writing dedicated java classes that complement the declarative artefact meta-model.

A Comprehensive Meta-level Support

AT recognizes the existence of *meta-activities and meta-processes*. For instance, Bardram (1998) defines a 'co-construction level' where subjects *collectively reconceptualise their activity*. If the scope of meta-activities is restricted to the dynamic evolution of the environment then only a meta-interface is needed, i.e., a set of meta-operations that can use all dynamic malleability techniques previously discussed. Following the idea that "CSCL is a socio-technical process which requires careful planning and preparation by both students and teachers" (Carell et al., 2005), Omega+ gives a broader definition to the meta-level and focus on *cooperative meta-processes* for: (1) designing the learning situation and customizing the CSCL system, (2) monitoring the learning process and dynamically evolving the CSCL system, (3) post-analysing learning process results for further improvement of the situation and CSCL system, (4) supporting the pedagogical development of teachers within a community of practitioners. Three of these cooperative activities (1, 3 and 4), mainly asynchronous, stay clearly outside the scope of Omega+ synchronous system. The proposed solution is to provide a broader *collaborative web platform dedicated to CSCL practice, evaluation, and dissemination*. This platform, called ESCOLE+, aims at hosting virtual communities of volunteer teachers, CSCL specialists, and students for designing, executing, and tutoring Omega+

based CSCL sessions, analysing them, and debating all related technical and pedagogical issues. The underlying open-source cooperative infrastructure developed in our research team (www.libresource.org) provides to its users a tree of projects and sub-projects, each project including a tree of documents and resources such as wiki pages, forums, issue trackers, mailing lists, news, download areas, surveys, versioning tools, user groups and roles, timelines (event lists), etc. End users can create, delete, move and modify resources and projects. Each project has its own security policy and access rights can be defined individually for each resource and role. New projects can be created resource by resource or by instantiating templates, i.e. predefined resource sub-trees. The generic modeller of Omega+ has also been customized for generating these templates from high-level visual models. ESCOLE+ provides three main spaces (projects): a 'Pedagogical Space', including a 'Community Space' for general information exchange and a 'Design Space' where Omega+ models are designed by teachers and CSCL specialists within dedicated sub projects, a 'Learning Space', where tutors and students execute model-driven collective learning processes within specialized sub projects, and a 'Platform Space', for managing users, groups, documentations and so on. This platform, still under development, complements Omega+ in three domains. First, ESCOLE+ provides web support for hybrid processes mixing synchronous and asynchronous activities, like other systems such as KnowledgeForum or Synergia. Secondly, ESCOLE+ centralizes detailed usage information, through Omega+ logs and ESCOLE+ event lists, making possible usage analysis for long periods of time and for different contexts. Finally, ESCOLE+ can support the pedagogical development of teachers within a community of practitioners: newcomers can learn by observing ongoing processes (in a similar way of what happens in open-source communities), by replaying recorded processes, by reading experiment reports and best practices catalogues, by communicating with CSCL specialists and other interested teachers. Later, observers can start to participate to collective learning activity definition and design. Finally, they can tutor activities with their own students or other students, possibly with the help of more experienced teachers at the beginning.

Conclusion

This paper proposes a set of high level requirements for designing-in-the-large realistic CSCL systems dealing with the general complexity of supporting people doing things collaboratively through computers and the specific complexity of creating pedagogical situations in which collaborative learning processes are expected to occur. A reflective architecture, providing definitional malleability, is complemented by mechanisms for operational and developmental malleability. Specialized modelling approaches, high-level visual modelling languages and cooperative meta-level support are other basic ingredients required for allowing teachers, who are not computer experts, to perform themselves customization, evolution, and improvement meta-activities. However, new research efforts following an iterative and experimental design process, will be needed for designing-in-the-small the rich and malleable systems the CSCL community is asking for.

References

Bardram, J. (1997) Plans as Situated Action: an Activity Theory Approach to Workflow Systems. In *Proceedings of the 5th European Conference on Computer Supported Cooperative Work (ECSCW'97)*, Lancaster, UK, Kluwer Academic Publishers, 17-32.

Bardram, J. (1998). Designing for the dynamics of cooperative work activities. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'98)*, Seattle, Washington, ACM Press, 89-98.

Carell, A., Herrmann, T., Kienle, A. & Menold, N. (2005). Improving the Coordination of Collaborative Learning with Process Models. In *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL'05)*, Taipei, Taiwan, 18-27.

Jermann, P., Soller, A. & Müehlenbrock, M. (2001). From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. In *Proceedings of Euro-CSCL*, Maastricht, The Netherlands, 324-331.

Jonassen, D.H. & Rohrer-Murphy,L. (1999). Activity Theory as a Framework for Designing Constructivist Learning Environments, *Educational Technology, Research and Development*, 47(1), 61-79.

Lonchamp, J. (2006). Supporting synchronous collaborative learning: a generic, multi-dimensional model. *Int. Journal of CSCL*, 1 (2), 247-276.

O'Donnell, A. & Dansereau, D. (1992). Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance, in Hertz-Lazarowitz, R., Miller, N. (eds.): *Interaction in cooperative groups - the theoretical anatomy of group learning*, Cambridge University Press, 120-141.

Suthers, D. & Xu, J. (2002). Kukakuka: An Online Environment for Artifact-Centered Discourse. *Education Track of the WWW Conference*, Honolulu, 472-480.