DOI 10.1007/s11412-006-8896-x

Supporting synchronous collaborative learning: A generic, multi-dimensional model

Jacques Lonchamp

Received: 00 Month 0000 / Revised: 00 Month 0000
Q1 Accepted: 00 Month 0000

Abstract Future CSCL technologies are described by the community as flexible, tailorable, negotiable, and appropriate for various collaborative settings, conditions and contexts. This paper describes the key design issues of a generic synchronous collaborative learning environment, called Omega+. In this approach, model-based generalizing is applied to the four dimensions of collaborative learning: the situation, the interaction, the process, and the way of monitoring individual and group performance. These four aspects are explicitly specified in a set of models that serve as parameters for the generic environment. This opens the possibility of combining many structuring/scaffolding techniques that have been proposed in isolation in the CSCL literature. The paper also emphasizes the specificities and difficulties of evaluating a comprehensive generic support approach. Experimental evaluations conducted by system designers generally isolate the effects of a particular design feature on learning. This kind of evaluation can hardly demonstrate the usefulness of a generic model at the global level and the feasibility of system customization by non-specialist teachers. To address these difficulties, Omega+ is integrated into a larger collaborative web platform dedicated to CSCL practice, evaluation (by collecting anonymized logs), and dissemination (by supporting the technical and pedagogical development of teachers).

Keywords CSCL · Synchronous learning · Model-based genericity · Interaction model · Process model · Artifact model · Effect model · Evaluation · Dissemination

Introduction

Computer-Supported Collaborative Learning (CSCL) aims at producing tools and environments for supporting collaborative learning, developing our understanding of learning processes, and finding the best ways to implement new approaches and tools into actual educational systems (Dimitracopoulou, 2005). Despite the

J. Lonchamp (✉)
LORIA—Université Nancy 2, BP 239, Vandœuvre-lès-Nancy, 54506, France
e-mail: jloncham@loria.fr

AUTHOR QUERY

AUTHOR PLEASE ANSWER QUERY.

Q1. Please provide history dates.

production of an impressive number of tools and environments by the CSCL community during its first decade, CSCL adoption remains slow and challenging when compared to the dissemination of more classical e-learning environments that support instructionalist pedagogy (Haatainen & Korhonen, 2002).

Our analysis of synchronous CSCL systems in Section 2 highlights one of the possible reasons for this problem: most existing systems suffer from *restricted applicability* (limiting re-use), in the sense that they are characterized by very specific situations, particular forms of interaction and specific learning processes. So far, researchers have focused primarily on the most important issues of collaborative learning *in isolation* and have proposed and evaluated *highly specialized systems*. Conversely, future mature CSCL technologies are described by the community as *richer* and *appropriate for various collaborative settings, conditions and contexts* (Dimitracopoulou, 2005), *reconfigurable, adaptive*, offering *collections of affordances and flexible forms of guidance* (Suthers, 2005), and *very flexible and tailorable* (Lipponen, 2002). Our analysis of synchronous CSCL systems shows the beginning of a trend towards systems with a *larger applicability and flexibility* during the last 5 years. Our aim is to go one step further in that direction by providing a *generic environment*, called Omega+, that is not tightly tied to some specific usage situation, learning process or knowledge type. Teachers can *fine tune* this generic environment to meet their specific requirements and pedagogical strategies. The "irreducible kernel" corresponds to a regular *chat tool* while richer configurations support *flexible combinations of facilities for structured textual communication, scripted collaboration, and collaborative construction and manipulation of shared artifacts*. In this introduction we briefly discuss our choices from three points of view: *architectural, theoretical, and evaluative*.

From *the architectural viewpoint*, neither monolithic integrated environments that include a large and flat collection of predefined mechanisms in a 'Swiss army knife' fashion, nor loosely integrated tools at the presentation level can realistically meet the requirements for larger applicability and flexibility. Component-oriented solutions, where each component adheres to a common specification and implements a given functionality that can be composed with others, could be more effective. We have chosen a different approach, however, called *model-based genericity*, in which the system behavior depends of the interpretation of some explicit model. We chose this approach because this solution provides *finer grain customization capabilities* than the component-based approach. IMS Learning Design (LD) players, such as the RELOAD LD player (http://www.reload.ac.uk/ldplayer.html) and Edubox LD player (Tattersall, Vogten, & Hermans, 2005), are well-known examples of model-based generic systems in the e-learning field. IMS LD is able to describe units of learning based on different theories and models of learning together with the learning objects used, and can be adjusted to personal needs. The generic player can scaffold the learning process in accordance with the IMS LD model currently loaded. A frequent drawback of model-based genericity is the overwhelming complexity of the meta-model, which defines all the concept types of the modeling language and, as a consequence, the complexity of models defining possible behaviors for the system. The conceptual model of IMS LD is a good example, which includes more than 40 concept and relationship types, while lacking adequate elements for modeling collaborative activities (Miao, Hoeksema, Hoppe, & Harrer, 2005; Hernandez, Asensio, & Dimitriadis, 2004). Such complexity makes model engineering a challenge for teachers who usually are not experts in computer science specification. Reuse of large models is also problematic because they have a high probability of including some inappropriate elements. In our approach, called *multi-dimensional model-based genericity*, we propose to use *separate models* for each dimension of collaborative learning. Following Dillenbourg (1999), we consider four dimensions: (1) the collaborative *situation*, including, in particular, the kind of *artifacts* that are manipulated; (2) *the interactions* that take place within the participants; (3) *the learning process*; and (4) the set of *effects* in terms of individual and group performance. Separately, each model—process model, protocol model, artifact model, effect model—is quite simple, but as a whole the four models are sufficient for representing a wide range of pedagogical settings and contexts. In addition, when the basic models are defined, a small number of additional mechanisms can be selected at instantiation, as long as they are consistent with the previous choices. Section 3 details these key architectural issues and emphasizes how the generic environment favors visual modeling for non-specialists and model reuse.

From the theoretical point of view, the approach aims at scaffolding learners in complex synchronous tasks. The term *scaffolding* comes from the works of Wood, Bruner, and Ross (1976). The term was developed as a metaphor to describe the type of assistance offered by a teacher or more knowledgeable peer to support learning, altering the learning task so the learner can solve problems or perform tasks that would otherwise be out of reach. Scaffolding is associated with Vygotsky's notion of the zone of proximal development, which characterizes the region of tasks between what the learner could accomplish alone and what he or she could accomplish and master with assistance (Vygotsky, 1978). When the learner takes responsibility for or masters the task, the teacher begins the process of *fading*, or the gradual removal of the scaffolding, which allows the learner to work independently. *Software scaffolding* provides some sort of *structure* that helps make the learning more tractable for learners. Concretely, this can be done in a lot of different ways: by providing constrained or typed messages, restricted interaction protocols, pre-defined processes (scripts), structured workspaces, shared graphical representations like concept maps, graphical argumentations, disciplinary representations and simulation models. All these structuring techniques have been proposed and evaluated *in isolation*. To our knowledge, *combinations of structuring techniques* within the same learning activity have never been analyzed systematically. However, interactions do exist between these different aspects. For instance Löhner, van Joolingen, and Savelsbergh (2003) and Zhang (1997) demonstrate, in different application domains, that representations guide, constrain and even determine processes. Our system *opens the possibility to test a large number of combinations of structuring techniques*. Moreover, structural constraints also pose the problem of the *degree of coercion*, i.e., the degree of freedom that the learners have in following them. “Choosing the appropriate level of coercion is the oldest educational design trade-off. A certain degree of coercion is required for efficiency reasons, but too much might be in contradiction with the very idea of collaborative learning and might decrease student motivation” (Dillenbourg, 2002, p. 20). In software scaffolding, structural constraints *are enforced* (imposed) by the system. “*Definitional malleability*” allows designers to include in the models *a selected number of structural constraintsstatically* (i.e., before the beginning of the learning process) such as process rules (precedence rules), protocol rules (adjacency pairs of utterance types), ontologies of concepts in graphical shared artifacts, and ontologies of speech acts or sentences openers. As a complement to such definitional malleability, tutors and students need run-time flexibility, which we call "*operational malleability*." End users should be able to *change the model*, that is, modify the set of selected structural constraints, during the model-driven learning process. End users should be able also to *relax or sidestep* most of the structural constraints that apply, *without model evolution*, when exceptional circumstances arise. In this last case, the system should be in charge of making other end users aware of these punctual rule breakings. Operational malleability answers the "situatedness" of human learning and action (Suchman, 1987; Winograd & Flores, 1986). The s*ituated learning* theory sees the teacher's role—observing students, offering hints and reminders, providing feedback, scaffolding and fading, modeling, and so on—as integral to the learning situation (Herrington & Oliver, 1995). Collins, Brown, and Newman (1989) point out that this work is highly situation-specific and is related to problems that arise as students attempt to integrate skills and knowledge. Operational malleability is the primary mechanism for permitting tutors to take into account such *contextual and situational* elements. It is worth noting that definitional malleability is also important for taking into account *contextual elements* known from the beginning of the process. For instance, the teacher may wish to link the concept to be learned with something learners already know, or to define situations, processes and representations reflecting learners' cultural and social norms.

From the evaluation viewpoint, it is important to define who the stakeholders are, what the object of the evaluation is, and how the technology will be judged (Holst, 2000). In what follows, we mainly consider three types of stakeholders: *learners*, *teachers* (who design and tutor collaborative learning sessions), and *developers* (who build software). First, for *learners*, classical experimental evaluations are not well adapted for such a comprehensive environment because *it is meaningless to isolate the effects of a particular design feature on learning*. One possible way of evaluating Omega+ is to verify that the generic system makes it possible to bring *the same kind of support* as do existing specialized tools. So, we indirectly rely on the studies that evaluate the different techniques in isolation (summarized in Section 2) and expect to obtain interesting results when combinations of structuring techniques come into play. For demonstrating the equivalence with simple tools, like Belevedere's support for building inquiry maps (Suthers & Jones, 1997), a comparison of the provided functionalities is sufficient. For more complex tools, we make use of full-fledged *scenarios*. Scenarios include protagonists with individual goals or objectives and reflect exemplary sequences of actions and events when using the environment. They can refer to observable behavior as well as mental processes, and can cover situational details assumed to affect the course of actions (Rosson & Carroll, 2002). They might explicitly refer to the underlying culture, norms, and values (Bødker & Christiansen, 1997). They focus on specific situations, only enlighten some important aspects, and generally do not include every eventuality (Benner, Feather, Johnson, & Zorman, 1993). Beside their classical use in the design process, scenarios are used here for purposes of *evaluating the concrete benefits for learners of using some customized version of the generic environment*. In Section 3, all examples are taken from our set of evaluation scenarios, such as the implementation of the explanation protocol described in Pfister and Mühlpfordt (2002).

From the point of view of *teachers*, we aim at demonstrating that they can play an active role in the customization of the environment, both during the design phase by participating in the defining of models, and at execution time by adapting the system while tutoring learning processes. In this perspective, many questions can only be answered *by using the environment in a variety of real-life settings*: Is modeling doable by teachers? What percentages of teachers try to customize library models? Which models are chosen in the library? How and which dynamic malleability capabilities are used by tutors and learners? Our environment, Omega+, is a follow-up of a previous and more restricted tool called Omega Chat, which provided only process and protocol-oriented genericity (Lonchamp, 2005). The evaluation strategy for Omega Chat was to deliver the tool as an open-source product and to wait for feedback from users. Download statistics suggest there were more than a hundred downloads in 12 months from Sourceforge repository (http://omegachat. sourceforge.net). But most feedback messages were from developers raising technical issues—how to integrate the tool within their own systems, for instance—and not from end users. Our new strategy for obtaining more feedback and for answering usage questions *is to provide end users with collaborative web platforms dedicated to CSCL practice, evaluation and dissemination*. We have developed such a platform around Omega+ for a virtual community of teachers, students, and CSCL researchers who (1) execute model-driven collaborative learning processes; (2) design and develop the associated models; (3) analyze past activities and (4) cooperate through community tools like forums, mailing lists, wikis, issue trackers and document repositories. Anonymized logs will be collected for all experiments performed through these platforms. Section 5 summarizes this technological support dedicated to CSCL practice, evaluation and dissemination. Finally, *software developers* are also important stakeholders. In some circumstances *programmatic extensions* to the generic environment cannot be avoided. Section 4 discusses "*developmental malleability*" and illustrates how some aspects of the Omega+ architecture help software developers to perform their task more efficiently.

The rest of the paper is organized as follows. Section 2 analyzes a representative sample of synchronous CSCL systems and emphasizes the recent trend towards systems with larger applicability and flexibility. Section 3 presents the key design issues of Omega+, our multi-dimensional generic environment. Section 4 discusses the approach and shows that definitional malleability provided by the approach is not sufficient and has to be complemented by operational and developmental malleability. Finally, Section 5 describes the collaborative web platform for supporting both Omega+ usage analysis and, more generally, CSCL dissemination far beyond the small kernel of early adopters.

A survey of synchronous CSCL systems

Classifications

Researchers have proposed different taxonomies of synchronous collaborative learning systems. One is based on the kind of collaborative activities that they each support. Dimitracopoulou (2005), distinguishes between *text-production oriented systems* and *action-oriented collaborative systems*, and proposes a new *mixed category* of richer collaborative learning environments. Another classification, based on the kind of feedback provided to users (Jermann, Soller, & Muehlenbrock, 2001), distinguishes between *mirroring systems*, which display basic actions to collaborators, *monitoring systems*, which represent the state of interaction via a set of key indicators, and *guiding systems*, which offer advice or guidance based on an automatic interpretation of those indicators. Our own taxonomy is complementary with the two previous ones and distinguishes between *simple tools/environments* and *generic model-centered systems* (i.e., using explicit models as parameters). In the next subsection we apply these three characterizations to a representative sample of synchronous CSCL systems that are frequently mentioned in the CSCL literature.

Synchronous CSCL systems analysis

Table 1 roughly preserves the chronology of the development of chat systems. Each line summarizes the main characteristics of a system in terms of the pedagogical situation considered, the provided interaction means, and the main questions evaluated by its designers.

In the first period, prior to the year 2000, most chat systems were *simple tools*, designed around *very specific predefined situations* (e.g., Covis, Coler, C-Chene, Comet, Algebra-Jam) and *particular forms of interaction and processes* (e.g., Dialab, Belvedere, Group Leader Tutor, Better Blether). Most of them are *mirroring* systems. The small number of advising systems were *domain-specific* (e.g., Coler, Group Leader Tutor).

In the second period of development during the five last years, *different kinds of genericity* appeared, mainly for defining the kind of artifact that is collaboratively manipulated (e.g., Models Creator, Modeling Spaces, Cool Modes, Dunes, Co-Lab), and less frequently the interaction dimension (e.g., ProChat, Learning Protocol, ACT) and the process dimension (e.g., LeadLine). In most cases, modeling does not require programming skills. However, in Cool Modes for instance, specifying artifact operational semantics requires that ad hoc Java classes be written and linked to the declarative artifact model. *In all cases, a single generic dimension is considered.*

The next section describes our approach, based on multi-dimensional genericity (situation, interaction, process, monitoring), that could constitute another step towards more tailorable and flexible systems.

Omega+ design approach

A Chat-oriented kernel

As demonstrated in the previous section, most CSCL systems include a regular or structured chat either as a *core functionality* (for text-production oriented systems) or as a *complementary communication channel* (for action-oriented systems). Omega+ environment is built around *a chat-oriented kernel*, providing the usual functionalities found in regular chat tools with multiple rooms and private channels (whispering).

We start by recalling some well-known deficiencies and limitations of regular chat tools (Garcia & Jacobs, 1999; O'Neil & Martin, 2003) because the *structural extensions in the process and protocol dimensions* of Omega+, which will be described in the next two sections, try to bring solutions to these problems. The most important one is the *lack of control over turn positioning*. Since turns can be posted simultaneously by a number of participants, there is no guarantee that a response to a question, for example, will appear directly after the question that elicited it.

Instead, other turns may appear between a question and its response, causing confusion about threading. The consequence is a preference for short turns so that the response might be closer to the question, if sent quickly. Standard chats are not places where carefully constructed messages can be sent. Lack of visibility *of turns-in-progress* (chat systems only transmit turns when they are completed [ENTER key]) and lack of visibility *of listening-in-progress* (participants do not receive moment-by-moment information about the reaction of those who are listening to them), are other examples of well-known issues. Many other problems are documented in the literature but we restrict the discussion here to coordination issues.

A number of research prototypes address these problems by providing *non-standard interfaces*, such as threaded interfaces (Smith, Cadiz, & Burkhalter, 2000), 2D/3D graphical interfaces (Kurlander, Skelly, & Salesin, 1996; Viegas & Donath, 1999), and streaming interfaces (Vronay, Smith, & Drucker, 1999). It has been observed that each solution can solve one specific problem but can often create new difficulties for end users in other domains (Vronay et al., 1999).

Other research approaches extend traditional chat tools with *additional awareness mechanisms*, each addressing a specific issue: turns-in-progress visualization, social presence via animated face icons representing facial expression, hand raising (Fadel & Nazareth, 2004), and social proxy (Bradner, Kellog, & Erickson, 1999). Some of them are now integrated into commercial tools, like the textual "someone is typing" indicator. However, in our opinion, such additional awareness mechanisms cannot be accumulated in a "Swiss army knife" fashion, but *should be selectively available within consistent interaction styles* for avoiding an excessive level of cognitive load. For instance, in a round-robin interaction, user activity, turns-in-progress, and hand raising cues are obviously of little value.

Finally, the last approach considers that all these deficiencies are consequences *of the unstructured nature of standard chat conversations*. By constraining the turn-taking (as in moderated chat systems) and by dividing discussions into more focused sub-discussions, most coherence and coordination problems could be alleviated.

We think that educational settings strongly require such *structuring capabilities* for reasons that go beyond the aforementioned coordination issues. We have already discussed in the introduction section the interest of software that *scaffolds learners in complex tasks*. The next two sections discuss the *structural extensions* to the Omega+ chat kernel in the interaction and process dimensions.

A generic extension in the interaction dimension

The general idea is to scaffold productive interaction by encompassing interaction rules in the medium (Dillenbourg, 1999).

In a first approach, researchers have identified collections of *conversational moves* that they believe are necessary for an effective learning dialogue, and have implemented these moves as mandatory sentence openers (or complete utterances) in what are called "semi-structured interfaces." C-Chene, Better Blether, Smart Chat, and OXEnTCHE-Chat are examples of systems with a fixed set of sentence openers. ACT is an example of a *generic* solution with a customizable set of sentence openers and speech acts. The true efficiency of "such semi-structured chats" remains an open issue (Baker, 1997).

t1.1 **Table 1** Analysis of a representative sample of synchronous CSCL tools

	System	Situation	Interaction	Evaluation
t1.2	System	Situation	Interaction	Evaluation
t1.3	Dialab (Moore, 1993)	Argumentation and critical thinking for two players	Structured chat with sentence openers + dialog rules + turn taking (rigid logic-based dialogue game)	Text-based communication can lead to misunderstanding because non-verbal and paralinguistic cues are not available.
t1.4	Covis (Pea, Edelson, & Gomez, 1994)	Scientific inquiry in atmospheric and environmental sciences	Visualization tools + desktop video conferencing + application sharing	Qualitative and quantitative studies concerning students' attitudes, criteria for "good projects," communication technologies, communication bandwidth, visualization tools usage, teachers' perspectives.
t1.5	Group Leader Tutor (McManus & Aiken, 1996)	Problem solving discussion between two students	Structured chat (sentence openers)	Evaluating the students' co-operative attitudes before and after using the tool, and evaluating the students' academic achievements.
t1.6	C-Chene (Baker & Lund, 1996)	Energy chains modelling	Structured chat (sentence openers + turn taking)	Comparison of chat-box and structured interface interactions.
t1.7	Belvedere (Suthers & Jones, 1997) (Suthers & Hundhausen, 2002)	Collaborative inquiry for a small group	Inquiry map (predefined discourse acts + evidential relations)	Impact of representation type >(matrix, graph, text) on students' elaborations of their emerging knowledge.
t1.8	Better Blether (Robertson, Good, & Pain, 1998)	Group discussion for primary school	Structured chat (sentence openers)	Evaluation of BetterBlether as a communication tool, based on a conversational analysis, and comparison to a previous study of supervised and unsupervised groups.
t1.9	Comet (Soller et al., 1999)	Software design problems (OMT)	Shared OMT Diagrammer + structured chat (sentence openers + speech acts)	Determining the effectiveness of knowledge sharing episodes.
	Algebra-Jam (Singley, Singh, Fairweather,	Algebraic problem solving	Blackboard (goal tree) + topical chat with typed messages	Study of the flow of information that occurred between

t1.10	Farrell, & Swerling, 2000)			teachers and students and the types of mediation.
t1.11	ProChat (Whitehead & Stotts, 2000)	Chat with an explicit collaboration model	Structured chat with coloured Petri Net protocol specification	–
t1.12	Lead Line (Farnham et al., 2000)	Scripted social interaction	Generic chat with XML scripts defining roles and scenes	Effect of a structured script on the groups' ability to achieve consensus and to make better decisions.
t1.13	Coler (Constantino-González & Suthers, 2001)	Entity-relationship modelling	Private/public workspace + chat	Study of how students used and evaluated the coach's advice.
t1.14	Models Creator v3 (Fidas, Komis, Avouris, & Dimitracopoulou, 2002) Modelling Space (Avouris et al., 2004)	Open modelling system (building models out of primitive objects + qualitative and semi quantitative relations)	Shared workspace + chat + editor of primitive object + supervision tool in Modelling Space	Impact of alternative protocols of locking of the common work surface in which the model is built.
t1.15	Cosar (Jaspers, Erkens, & Kanselaar, 2001)	Collaborative writing	Shared text editor + chat	Relations between tool and resource use frequencies and the scores for the argumentative quality of the texts.
t1.16	Learning Protocol (Pfister & Mühlpfordt, 2002)	Collaborative text-based discourse	Structured chat for scripted cooperation (learning protocol = typed messages + explicit message sequencing + roles + explicit references)	Comparison of an explanation discourse guided by the EXP-protocol with an equivalent discourse using conventional free-text chat.
t1.17	Drew (Baker et al., 2003)	Collaborative argument grapher	Argumentation graph + opinions on nodes and links	Comparison between chat with graph and chat-only. No significant difference.
	Cool Modes (Pinkwart, 2003)	Generic system for graphical modelling and discussion	Multiple representations through a plug-in system with explicit and pluggable domain semantics and AI	Plan to distribute a questionnaire to get an impression how skilled one has to be in order to a) use the system and b) develop plug-ins

Table 1 (continued)

	System	Situation	Interaction	Evaluation
t1.18			functionality (basic plug-ins for structured discourse,...) —XML models	for it.
t1.19	Dunes (Börding et al., 2003)	Argumentation based learning	Shared argumentative maps, chat + off line analysis tool	–
t1.20	Smart Chat (Siebra, Christ, Queiroz, Tedesco, & Barros, 2004)	Intelligent environment for collaborative discussions	Structured chat + argumentation model	–
t1.21	Bubble Chat (Münzer & Xiao, 2004)	Process-aware chat for professional training	Chat with process support + learning instructions & materials	Feedback data indicated a low acceptance of the software tool. Participants evaluated the process control as being restrictive.
t1.22	OXEnTCHE-Chat (Vieira et al., 2004)	Chat that monitors interaction for teachers and learners	Structured chat + automatic dialogue classifier for on-line feedback + chatterbot agent (automatic dialog coordinator)	Qualitative evaluation of the tool's usability and the quality of the feedback provided
t1.23	Mediated Chat v1 to v5 (Pimentel et al., 2005)	On line course debates	Extended chat tools (with threads, hard-coded protocols, message queuing, latecomer management...)	Individual evaluation of each mechanism independently (threaded chat, predefined protocols, and message queuing)
t1.24	Co-Lab (van Joolingen et al., 2005)	Collaborative scientific discovery learning environment	Tools for model sketching, model specification (system dynamics), model testing, communication (chat), knowledge sharing	Specific studies concerning learners' general impressions, working patterns with minimal guidance, comparisons with face to face work, utility of process scaffolding.
t1.25	ACT (Gogoulou et al., 2005)	Generic structured chat with monitoring facilities	Generic chat with SST (scaffolding sentence templates), roles, monitoring through threaded view	Empirical study of the opinions of students concerning the usage of a predetermined set of SST. Qualitative study of customisation facilities.

In a second approach, often called "structured chats," the interaction follows complex *protocols* with typed messages, role assignment, and message sequencing (e.g., ProChat, LearningProtocol, Mediated Chat). These interaction protocols are either hard-coded, like in Mediated Chat ("unique contribution," "unique contributor," "circular floor passing," "mediated debate") or explicitly specified by the users with a *protocol modeling language* (e.g., ProChat, LearningProtocol). Some evaluations are positive (Pfister & Mühlpfordt, 2002). Others emphasize a low impact of these protocols on chat confusion (Pimentel, Fuks, & Lucena, 2005) and a low acceptance from end users (Münzer & Xiao, 2004) in some circumstances.

None of these solutions are silver bullets, but they can play a positive role in some settings. Omega+ provides *both kinds of genericity*, as *mutually exclusive* solutions, corresponding to different levels of scaffolding. First, each time a room is instantiated, two additional structuring mechanisms are made available when they are compatible with the room definition:

– a *customizable set of sentence openers*
– an utterance numbering system that allows users *toreference* previously published messages.

Second, at room definition time, the designer can select *predefined* or *user-defined application-dependent interaction protocols*. Each protocol definition includes a set of roles, a set of typed messages (utterances), and a set of rules defining *adjacency pairs* (Clark & Schaefer, 1989): if a user playing the role A produces a message of type X then any user (or the next user in a circle) playing the role B can continue with a message of type Y. Application-specific protocols can be defined graphically in a tree (or forest) form. The root(s) is (are) the role(s) that can start the discussion. Leaves are actor types receiving a message type in a situation already described somewhere in the graph. Figure 1 shows the Explanation protocol model defined in Pfister and Mühlpfordt (2002) within the Omega+ generic editor when the user has selected the "ProtocolModel" type and the "Explanation" protocol model.

At each moment, a participant using the *protocol-driven chat* (see Fig. 2) can only select a type of message in accordance with his/her role and the protocol rules. Figure 1 shows that the interaction always starts with a Tutor giving an explanation. Jack is the Tutor in the example of Fig. 2. Then, any Learner can produce a Question, an Explanation or a Comment. In the example, Mary gives a Comment. After a Question from a Learner (Suzan in the example), a Tutor (Jack) can only answer with an Explanation. It is the only move proposed by the chat client (Fig. 2). After the Explanation, the model prescribes that the next move is from the next Learner in a circular order (see the quantifier property visible in the tool tip of Fig. 1). In this example, it would be Mary. As already specified in the tree, a Learner receiving an Explanation from a Tutor can continue with a Question, an Explanation or a Comment.

A generic extension in the process dimension

Explicit script models are commonplace in asynchronous CSCL systems. For Dillenbourg (2002, p. 11),

> A script is a story or scenario that the students and tutors have to play, as actors play a movie script. Most scripts are sequential: students go through a

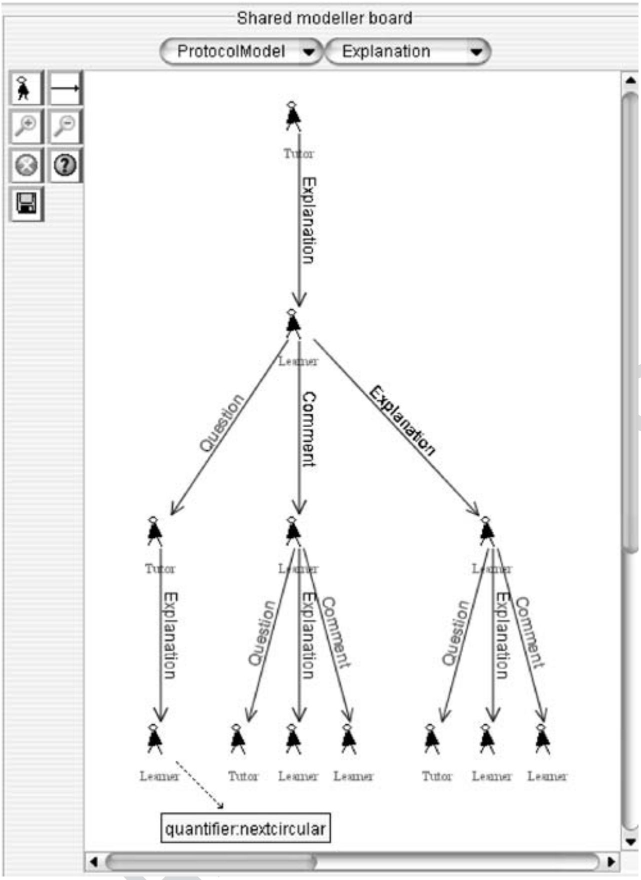

Fig. 1 A protocol model

> linear sequence of phases. Some scripts are defined in an iterative way, but from the student's point of view, they are run as a linear sequence. Each phase of the script specifies how students should collaborate and solve the problem. This requires five attributes: the task that students have to perform, the composition of the group, the way that the task is distributed within and among groups, the mode of interaction and the timing of the phase.

Among synchronous CSCL systems, some tools include predefined hard-coded processes and a few of them accept explicit script models as parameters (e.g., Bubble Chat, Lead Line).

Omega+ supports *process model-drivenexecution* and provides *process modeling facilities*. A process model, within a "structured room," defines a sequence of phase types. We differentiate between "regular" and "split" phases. In a regular phase the whole group of participants works in the same room. A *split phase* is a structured phase comprising a set of sub-phases running in parallel. The group of participants is divided into sub-groups working in different sub-rooms. Room Operators partici-

Fig. 2 The corresponding protocol-driven chat tool

pate in all sub-rooms. All sub-phases of a split phase start and terminate simultaneously. Each phase type (regular or sub-phase) is characterized by a name, a type (regular or split), an informal description, an interaction protocol type, and a set of available tools (at most 3). Some protocol types are predefined ("open-floor," "moderated open-floor," "circular floor passing," "single contribution," or "unique contributor"), while others are application-specific (see Section 3.2). A library of predefined process models is available for re-use at room definition time.

When a phase instance is created, the Room Operator:

- gives a name (by default the type name with an instance number),
- defines who is participating (if the phase has restricted participation) and the binding of users to protocol-specific roles (e.g., who is the Moderator in a moderated phase),
- gives some informal instructions,
- when it is compatible with the room type and useful, customizes all chat clients with sentence openers and/or explicit referencing (see Section 3.2).

The execution of a process model is flexible. The simplest execution just follows the predefined sequence by using the Next button. But Room Operators can also *jump from one phase to another* (e.g., skip a phase or iterate to a previous phase) by using the Jump button. In Fig. 2, Jack plays the application-dependent role of Tutor and the predefined role of Room Operator. So, the Next and Jump buttons for navigating in the process are visible in Jack's client. We will see later, in Section 4, that the process model structure can also evolve dynamically at execution time.

Figure 3 shows the Omega+ generic editor when the user has selected the "Process Model" type and the "Brainstorming" process model. This process model is just a sequence of three phase types: "Present" (where only the Room Operator can speak to describe the problem), "Propose" (where participants freely propose

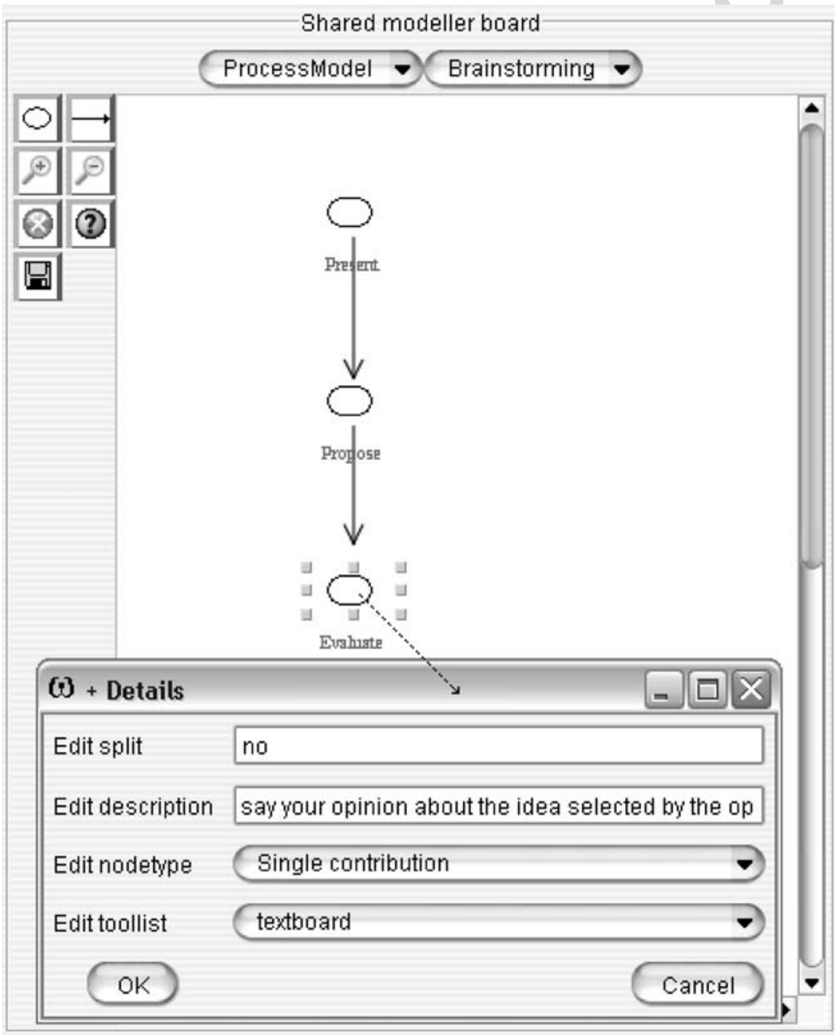

Fig. 3 The Brainstorming process model

ideas) and "Evaluate" (where each participant must give a personal opinion about the selected idea once—single contribution protocol). This last phase is iterated for each idea by the Room Operator. Another process model could include a split phase including two sub-phases ("Design1" and "Design2"). During each of them, a subgroup of participants build a UML model for a problem presented during the "Introduction" phase. During the "Integration" phase, the reunified group compares, evaluates and integrates the two proposals.

A generic extension in the situation dimension

Sharing and commenting on resources and material is another basic functionality of many CSCL environments that often complements textual communication. Several recent systems offer *generic means for defining these shared artifacts* (e.g., Modeling Space, Cool Modes, Dunes).

Omega+ provides:

(1) different kinds of shared conceptual representations: text, drawing, image, and *user-defined disciplinary representations*, taking the form of *graph-based notations*; thanks to this last feature, the system can support ad hoc notations conforming to a variety of social and cultural norms;
(2) rooms with several editors for different representations (e.g., a shared text editor for writing down preliminary ideas, a shared whiteboard for informal sketching, and a specialized diagrammer for formalizing a design);
(3) rooms with several instances of the same tool (e.g., for comparing and merging different views);
(4) import/export capabilities from one editor to another (e.g., exporting a text or a domain-specific graph into a whiteboard for freehand commenting).

The complete specification of a disciplinary graph-based representation should include the *visual representation of nodes and edges*, *integrity conditions* restricting the possible structures, and the *operational semantics* attached to the graph. Most generic systems only consider the first two aspects (e.g., Modeling Spaces, Dunes). In Cool Modes, all aspects are defined in XML "Reference Frame" files. However, the treatment of rules that contain domain specific operational semantics is implemented through a link to a dedicated Java class (Pinkwart, 2003). Our choice is to keep the same kind of *visual specification* for artifact meta-models as for protocol and process models, excluding complex operational semantics specification. The idea is to support with the same generic visual editor *the collaborative construction of new formalisms by teachers*, and *the collaborative construction of artifacts based on these formalisms by students*. If some specific behavior is needed we suggest using a display sharing external dedicated tools or to build *ad hoc rooms* (see Section 3.6).

Figure 4 shows Omega+ generic editor when the user has selected the "Diagram Model" type and the "State Diagram" meta-model. Icons on the right part of the screen specify available node and edge types with their attributes (button icon for the tool palette of the generated state diagram editor, component icon of state diagram node types, color, line and arrow style of state diagram edge types). The graph on the right part of the screen shows which are the allowed connections between all these structural elements (start node to transition edge to state node; start node to transition edge to stop node; state node to transition edge to state node; state node to transition edge to stop node).

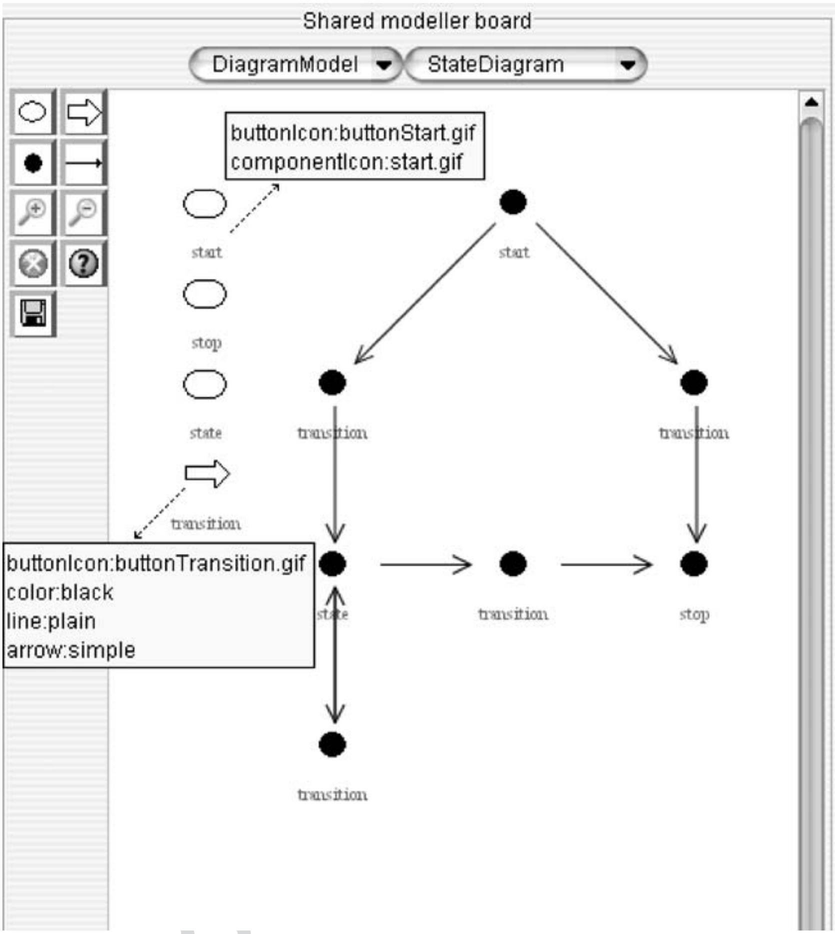

Fig. 4 An artifact model

Figure 5 shows the generic editor configured by the "State Diagram" meta-model. Some built-in facilities are provided in this instance of the generic editor, such as the refinement of a node into a sub-graph (eighth and ninth buttons in the tool palette). Refined nodes have a visual cue for identifying them (a filled square box, as shown into the "running" state of Fig. 5). The refinement level of the current graph is written in the top bar.

A generic extension in the effect monitoring dimension

According to Barros and Verdejo (2000), at a first level called the *performance level*, users' actions are observed and recorded. *Mirroring systems* automatically collect rough data about students' actions and reflect this information back to them (Jermann et al., 2005). At a second level, called the *analysis level*, some indicators

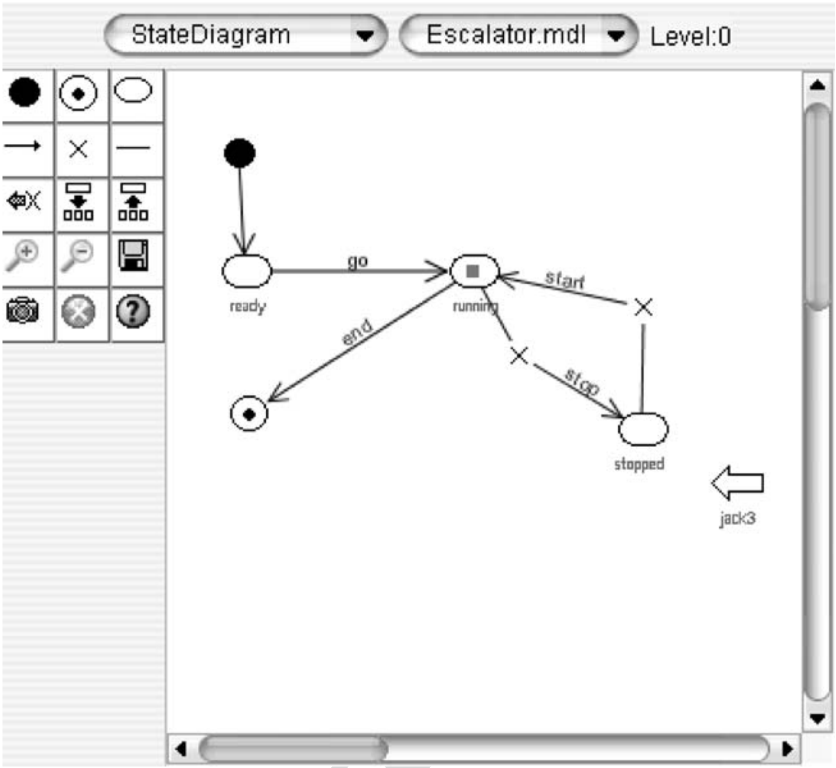

Fig. 5 The corresponding state diagram editor

characterizing users' actions are computed. *Monitoring (or meta-cognitive) systems* display information about what the desired behavior might look like alongside a visualization of the current state of these indicators. It is then up to the students or teachers to interpret the views and decide what actions (if any) to take. In *advising/guiding systems* these attributes are automatically compared to the desired behavior by meta-cognitive agents.

In Omega+, *the process of constructing high level visualizations from a set of predefined low-level variables is generic*. An "effect model" describes the kind of results required for a specific learning process. An XML-based effect model file can produce several corresponding graphical views. An effect model specifies:

– general parameters, such as the time interval between measures for time series,
– the characteristics of each visualization to display: name, informal description, type (bar chart, time series), value labels, and expressions defining how values are computed from the low-level predefined variables.

In the current version of Omega+, graphical visualizations are restricted to stacked bar charts computed for each user and stacked time series. Values that are displayed are simple arithmetical combinations of the predefined low-level variables. Richer operators (e.g., square root, mean, min/max, etc.) should be provided in future releases.

Specific guidance through ad hoc rooms

Visual modeling of artifacts comes to its limits when the specification of complex or application-dependent behaviors is required. For teaching science, "computer modeling" has been recognized as an important approach (Spector, 2000) where students create their own *executable* external representations of a domain or subject. They can *simulate* the models they create and observe and draw conclusions based on the model output. In Omega+, we suggest to distinguish the collaborative design of the model by using a customized instance of the generic diagrammer and the animation of the model. For animating representations having standard operational semantics (like Petri Nets or System Dynamics) it is possible to use external dedicated tools through some application-sharing facility. For *animating representations with a non-standard operational semantics*, like in Model-It (Jackson, Stratford, Krajcik, & Soloway, 1996), or for *integrating different representations*, like in SimQuest (Löhner et al., 2003), the development of *ad hoc rooms* cannot be avoided. As a demonstrative example of both cases, we have implemented two specific argumentation rooms by reusing the approach from the European SCALE project: the Drew graphical argumentation tool (Baker, Quignard, Lund, & Séjourné, 2003), the Alex structured chat, and Alex–Drew integration. We do not describe here the rationale behind this approach but focus on the way ad hoc rooms can be constructed and integrated into Omega+. The *graphical argumentation room* is similar to the Drew argumentation tool: users directly manipulate graphical boxes and links. Students are able to express their opinions—"in favor" and "against"—about any element of the argument graph. In order to highlight differences of opinion, and to focus discussion upon them, *boxes in whichopposing opinions have been expressed appear in a "crushed" form* (see Fig. 6). This exemplifies a complex behavior difficult to specify in a generic declarative way. The *textual argumentation room* is similar to Alex-Drew integration: the two optional functionalities of the Omega+ chat kernel—sentence openers and explicit references between messages—are used for textually creating and linking arguments, similar to the Alex structured chat. The system automatically translates each utterance into a non-editable graphical view similar to those manually constructed in the graphical argumentation room. This kind of complex representation integration is also very difficult to specify in a generic declarative approach. It is worth noting that large pieces of code, both at the kernel and interface levels, can be re-used when building ad hoc rooms. Omega+ allows the programmer to *integrate these ad hoc rooms into every structured learning process*, like any other kind of room.

Discussion

Definitional malleability

Omega+'s basic orientation is multi-dimensional genericity. The previous section shows how most aspects of collaborative learning, i.e., the situation, the interaction, the process, and effects monitoring, are specified explicitly through a set of models that serve as parameters for the generic environment. In this way, *definitional malleability* is provided and most of the design trade-offs defined in Dimitracopoulou (2005) do not receive inflexible hard-coded answers, but adaptive and contextual

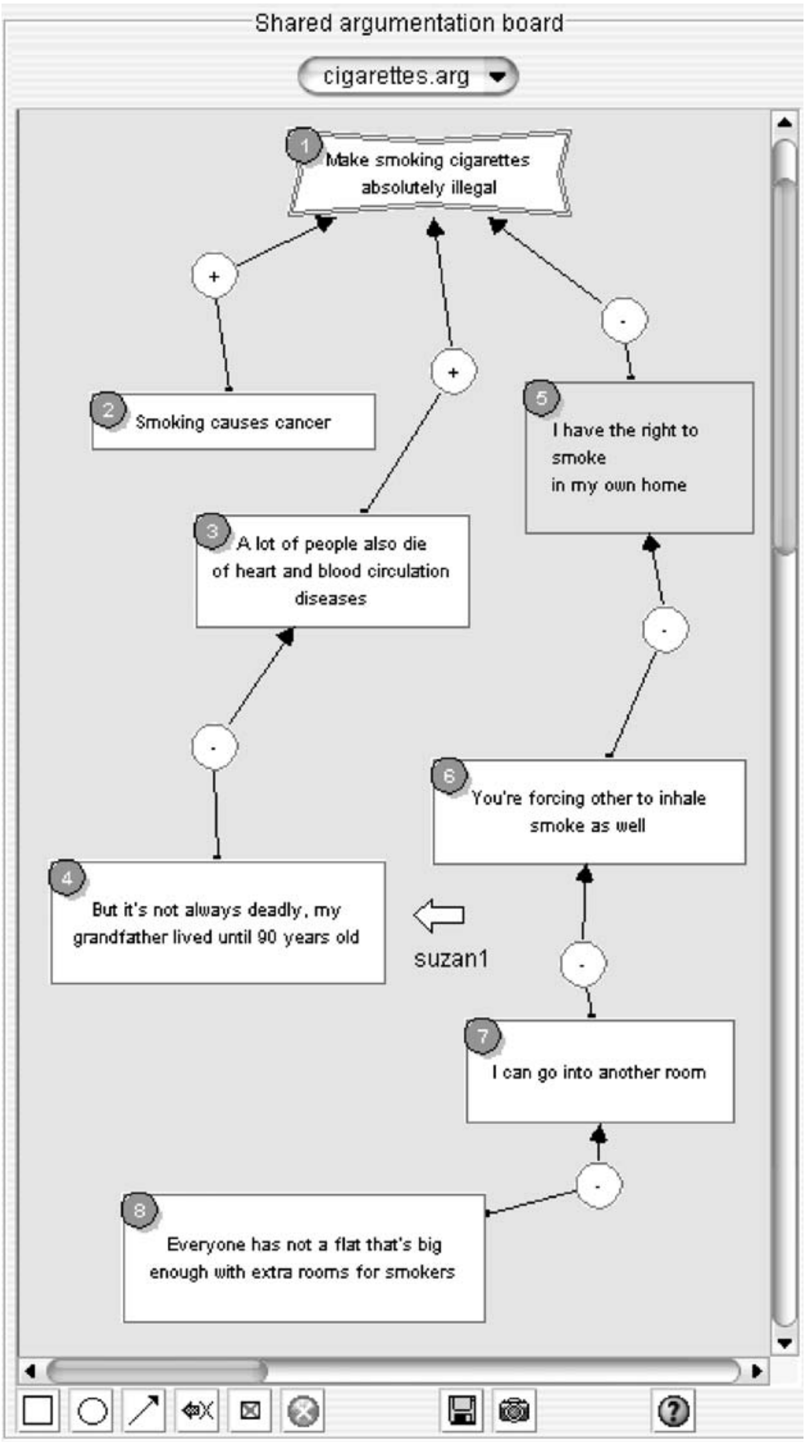

Fig. 6 A graphical argumentation room

ones: trade-offs between free and structured dialogue, trade-offs between restricted collaboration protocols vs. free ones, trade-offs between self-regulation (through meta-cognitive visualizations) and teacher support, trade-offs between parallel and embedded representations and tools for dialogue. The environment can be fine tuned *for various different collaborative settings, conditions and contexts*, by including in the models a selected number of structural constraints statically, i.e., before the beginning of the learning process: phase precedence rules, protocol rules, ontologies of concept types in artifact models, ontologies of sentences openers, etc.

The multiplicity of models brings some obvious potential advantages. Each (meta) model is simpler. Visual (graph-based) representations can favor involvement of teachers who can also more easily browse libraries of predefined models. On the other hand, this orientation also raises a number of questions because there are *many potential interactions across the four dimensions*. We have seen that interaction models are contained in process models as attributes, and that each phase in the process model can include tools parameterized by different artifact models. Figure 7 summarizes *the overall logical configuration process* of the generic environment with all decisions made at definition and instantiation times.

Some interactions are more subtle, and the way they are managed in Omega+ could be improved. Interactions between protocol models and artifact models raise some issues, such as the impact of a circular (round robin) protocol on the access rights of shared artifacts (floor management of shared editors). Another example of possible improvement can be found when considering the above-mentioned trade-off between *parallel and embedded* representations and tools for dialogue. Embedded solutions directly attach comments on the artifact under discussion, while parallel solutions use separated windows. Both solutions are available in Omega+. First, model designers can add *annotation node types* within any artifact model, the instances of which can be created and modified by end users (embedded solution). The chat tool can also be used as the medium for commenting on the artifacts (parallel solution). In parallel solutions, *explicit referencing* is often proposed for *reducing the distance* between the object of the discussion and the corresponding dialogue. Currently, Omega+ kernel provides two independent built-in facilities for explicit referencing:

– explicit references between messages (through message numbering),
– explicit references between messages and graphical representations (through "graphical pointers"). The whiteboard and all graphical editors provide a "pointer button." After this button has been clicked, an arrow with the user's name and a sequence number follows the mouse pointer and is drawn when and where the mouse is clicked (see Figs. 5 and 6). Chat productions can include explicit references to these personalized and numbered pointers.

These two referencing facilities are managed independently. They could be merged into some higher-level mechanism in the spirit of what is proposed in Mühlpfordt and Wessner (2005), for instance.

Operational malleability

Definitional malleability is not sufficient. Beside static flexibility, end-users need dynamic (run-time) flexibility, which we call *operational malleability*.

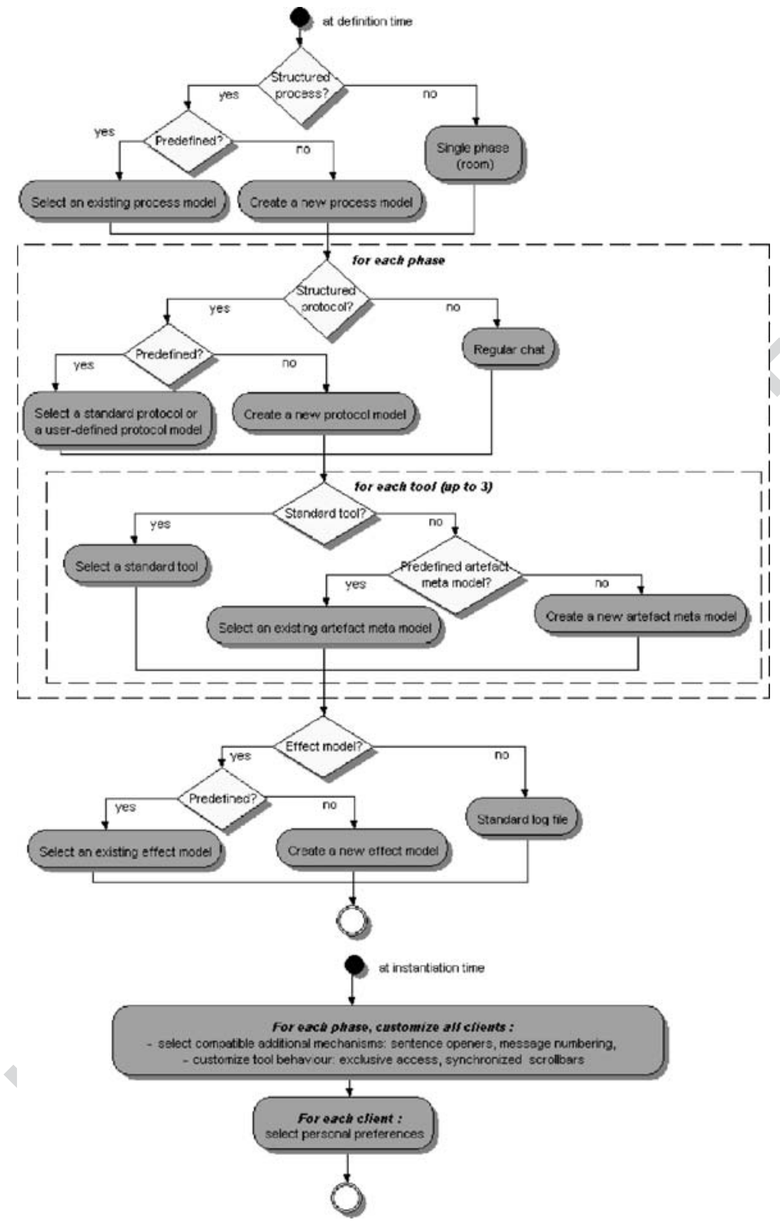

Fig. 7 The main configuration decisions at definition and instantiation time

First, users playing the predefined Room Operator role can *change the current models* during the model-driven learning process. For the currently executing process model, it is possible in a very simple way through *dedicated menus* (the template model remaining unchanged). Room operators can create, delete, and modify all phase types of the current model, at the exception of the phase type currently in execution (in this specific case, a new phase type has to be created and the operator has to jump to a new instance of this new type). Dynamic evolution of protocol and artifact models are expected to be less frequent than process model evolutions, and no dedicated menus are provided. The Room Operator must use the shared model editor in a model design room for creating new versions of these models, which can serve for changing the executing process model.

Second, room operators can *relax* or *sidestep most of the* constraints that apply when exceptional circumstances arise. The system is then in charge of making other users aware of these punctual rule breakings. Here are some examples available through contextual menus or buttons:

- skip a user during a circular (round robin) interaction protocol,
- kick a participant off for a given duration,
- jump to any another phase, before or after the current one,
- transfer the Room Operator role to any other room participant,
- transfer the Moderator role (predefined role in moderated rooms) to any other room participant.

Developmental malleability

Operational malleability concerns *end-users*, i.e., teachers and possibly students. Another kind of malleability, *developmental malleability*, concerns *tool developers*. The architecture of Omega+ aims at facilitating some evolutions of the implementation. Most properties are stored in XML files at three different levels.

- At a first level, the process meta-model, the protocol meta-model, the effect meta-model, and the artifact meta-meta-model describe in each dimension how the generic shared model editor (i.e., Omega+ design environment) has to be configured: what are the available node types, their required attributes, their visual properties; what are the available relation types, their connection constraints, and their visual properties.
- At a second level, process models, protocol models, effect models and artifact meta-models created with the Omega+ design environment are stored (e.g., the Brainstorming process model of Fig. 3). The first three models serve as parameters to the Omega+ execution environment, including the chat kernel extended with a model-driven engine. Artifact meta-models (e.g., the State Diagram artifact meta-model of Fig. 4) configure the generic shared model editor for producing a customized diagrammer.

At a third level, shared artifacts (e.g., the Escalator state diagram of Fig. 5) are manipulated by students during the collaborative learning process with the customized diagrammer. Thanks to the effect model, high level effect visualizations can be displayed for end users.

As an example of implementation change, suppose that a developer wants to distinguish between artifact node types that can be refined by students using the diagrammer from artifact node types that cannot be refined. For implementing this new feature, the first step is to create a Boolean property “canBeRefined” for nodes in the artifact meta-meta-model at the first level. Then, the developer can give a true or false value to this property for each type of node in the artifact meta-model with the Omega+ design environment. Finally, a few lines of code within the generic editor can perform the test of the "canBeRefined" property value of a given node type each time a user tries to refine a node of this type with the node refinement button.

Evaluation issues

Our study of synchronous CSCL systems in Section 2 emphasizes how their impact and effectiveness on learning is evaluated (Table 1). For simple tools of the first period (prior to the year 2000), most studies try to *isolate the effects of the central design feature on learning*: comparison of interactions using a regular chat interface and the dedicated structured chat interface for C-Chene (Baker & Lund, 1996), evaluation of the impact of representation on students' elaborations of their emerging knowledge for Belvedere (Suthers & Hundhausen, 2002), evaluation of Comet's analyzer of collaborative episodes (Soller, Linton, Goodman, & Lesgold, 1999), evaluation and use of Coler's automatic coaching facility (Constantino-González & Suthers, 2001), evaluation of the impact of alternative protocols for locking the shared work space in Modeling Space (Avouris, Komis, Margaritis, & Fidas, 2004), etc. For the more complex and generic tools of the second period (during the five last years), different evaluation approaches have been proposed, but none of them is fully convincing. Many generic systems are evaluated through a single or a small number of *pilot studiesusing specific models*. Evaluations of Learning Protocol (Pfister & Mühlpfordt, 2002), BubbleChat (Münzer & Xiao, 2004), LeadLine (Farnham, Chesley, McGhee, Kawal, & Landau, 2000) and ACT (Gogoulou, Gouli, Grigoriadou, & Samarakou, 2005) are typical examples. This kind of approach fails to answer fundamental questions, such as those concerning *the global interest of genericity* or *the concrete feasibility of system customization by non-expert teachers*. In other cases, *qualitative evaluations by questionnaires* are used for trying to answer these fundamental questions. This is the case for Cool Modes (Pinkwart, 2003) and OXEnTCHE-Chat (Vieira, Teixeira, Timoteo, Tedesco, & Barros, 2004). Finally, some researchers honestly recognize the lack of a convincing approach for realistically evaluating complex environments: "as Co-Lab is a large comprehensive system, evaluation studies have had to focus on speciffc aspects of it, rather than evaluating the whole system" (van Joolingen, de Jong, Lazonder, Savelsbergh, & Manlove, 2005).

Our proposal *for evaluating the global interest and concrete feasibility of the Omega+ generic approach*, briefly discussed in the introduction section, is twofold. First, the usefulness of the approach is demonstrated by showing that *Omega+ can emulate, at least in their main functionalities, a large set of existing tools*. For simple tools, the demonstration is only based on the list of provided functionalities. For more complex tools, evaluation scenarios mimic published pilot studies with the original tools. Table 2 summarizes a first list of tools that Omega+ can emulate completely or in large part.

Second, the strategy for realistically evaluating Omega+ usage questions *is to provide a collaborative web platform dedicated to CSCL practice, evaluation, and dissemination.* The ESCOLE+ platform (Environment for Supporting COLlective Learning Enthusiasts) is built on top of LibreSource (http://www.libresource.org),

Table 2 Examples of CSCL Tools That Omega+ Can Emulate t2.1

Tools	Emulated functionalities	Not emulated	
			t2.2
Lead Line (Farnham et al., 2000)	Scripted chat with scripts defining roles and scenes		t2.3
Better Blether (Robertson et al., 1998)	Structured chat with predefined sentence openers		t2.4
Belvedere (Suthers & Jones, 1997)	Visual inquiry environment using maps with discourse acts and evidential relations		t2.5
ACT (Gogoulou et al., 2005)	Generic chat with scaffolding sentence templates	The threaded view	t2.6
Learning Protocol (Pfister & Mühlpfordt, 2002)	Protocol-constrained textual environment		t2.7
Modelling Spaces (Avouris et al., 2004)	Visual modelling environment with a shared workspace, a chat and an editor of primitive objects	The supervision tool	t2.8
Comet (Soller et al., 1999)	Shared OMT Diagrammer and structured chat (with sentence openers and speech acts)	The analyser of collaborative episodes	t2.9
Drew (Baker et al., 2003)	Interactive tools for graphical argumentation		t2.10
Coler (Constantino-González & Suthers, 2001)	Private/public workspace for entity-relationship modelling and chat	The personal coaching agent	t2.11

an open source J2EE collaborative web platform developed in our research team and already used in different production environments. ESCOLE+ aims at hosting virtual communities of volunteer teachers, CSCL specialists, and students for *designing*, *executing*, and *tutoring Omega+ based CSCL sessions*, *analyzing them*, and *debating all technical and pedagogical issues*. ESCOLE+ provides Design Spaces for developers to deliver Omega+ process models. Each definition space is a design sub-project, created from a standard template with an instantiation tool. In each definition space, teachers and CSCL specialists can access *Omega+ design environment for creating, browsing, and customizing Omega+ models*. They can also use various communication tools for discussing all related pedagogical and technical issues (news, forum, issue tracker, wiki page, mailing list, etc.), and share technical documentations, experience reports, and Omega+ log files in the download area.

For creating a specific execution space within the Learning Space, designers can work in the "LibreSource style" by manually creating a LibreSource template in the design space and by using the dedicated instantiation tool. They can also work in the "Omega+ style" by generating an XML project file from a *LibreSource process model* created with the Omega+ design environment. In this case, Omega+ generic editor is parameterized by the *LibreSource meta-model* that defines all concept types necessary for describing an execution space: sub-space nodes, resource nodes, user-group nodes, precedence and inclusion relationships.

In our first lab experiments of ESCOLE+, we designed a collective learning process that aims at improving the skill of students in understanding and summarizing a complex document. It includes five steps implemented through five sub-spaces:

1.) "Initialization": in this space the tutor uploads the text containing the knowledge to be acquired which will be analyzed by a small group of three to five students.
2.) "Initial Summary": in this space students receive a description of their first task, i.e., reading the text and producing a personal summary (20 lines maximum); they download the text and upload their initial summaries; when all summaries are downloaded the tutor can announce the next synchronous session to all students.
3.) "CSCL Session": in this space students use the Omega+ synchronous tool and follow the COTEXT method (O'Donnell & Dansereau, 1992; Pfister & Mühlpfordt, 2002). The text is divided into as many sections as there are students. Each section is associated with a two-step iteration (Omega+ process model). At the beginning of each iteration, the Summarizer role is taken by the next student.

   - a "Production step" where a student, playing the Summarizer learning role, produces a summary,
   - a "Review step" where the other students act as Commentators in accordance with the COTEXT protocol (Omega+ protocol model). Each Commentator produces a correction, a supplement or a comment. If a correction or supplement is provided, it is the Summarizer's turn to accept or reject the proposed contribution. If a comment is provided, it is the next Commentator's turn. This cycle repeats until no correction or supplement is given.

4.) "Final Summary": in this space students must upload their final summary of the document taking into account all that has been said in the previous collaborative step.
5.) "Assessment": in this space the tutor reads all initial and final summaries for producing the final evaluation report describing how the different students have improved their summaries through the collaborative phase and the COTEXT method.

As a conclusion to this glance at the ESCOLE+ platform, we emphasize its fundamental role complementing Omega+ in three domains.

1.) ESCOLE+ provides support for *hybrid processes mixing synchronous and asynchronous interactions*, like other recent systems such as KnowledgeForum (Scardamalia, 2003) or Synergia (Stahl, 2004).
2.) ESCOLE+ is the way *to collect detailed usage information from the real world*, through Omega+ anonymized logs and ESCOLE+ event lists. Such information may include the percentage of teachers who try to customize library models, the models that are chosen in the library, and the dynamic malleability features that are used by tutors and learners.
3.) ESCOLE+ *supports both teacher and student learning*. The technical and pedagogical development of teachers can be progressive. First, newcomers can learn about pedagogical, technical and practical issues directly, by observing ongoing processes, in a way similar to what is described in open-source

communities (von Krogh, Spaeth, & Lakhani, 2003). They can learn also indirectly by reading experiment reports and best practices catalogs and by communicating with CSCL specialists and other interested teachers. Later, observers can start to participate in 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

Building flexible, tailorable, and negotiable systems, appropriate for various collaborative settings, conditions and contexts is a central objective of the CSCL community.

Omega+ promotes the concept of *multi-dimensional model-based genericity* for reaching this goal. This approach mainly provides static *definitional malleability* through the inclusion in models of a selected number of structural constraints. But definitional malleability is not sufficient and has to be complemented by dynamic *operational malleability* for tutors and students and *developmental malleability* for tool developers. Two fundamental issues concerning the *way to evaluate such a large comprehensive system* and the *way to ensure thetechnical andpedagogical development of teachers* also receive an original technological answer through the ESCOLE+ specialized collaborative web platform.

The next period in our research work will be mainly devoted to *enlarging the collection of predefined process models*. Each of them will be tested through lab experiments. We anticipate the fact that most teachers will probably give priority to predefined process models as defined at the ESCOLE+ level, including Omega+ synchronous sessions driven by predefined process, protocol, artifact and effect models.

The long-term objective of our research is to *enlarge the community of CSCL practitioners far beyond the current kernel of early adopters*. But we are aware that we still have a long way to go to build truly mature CSCL environments of the next generation, which more powerful and flexible. The example of open-source software shows that *a large exposure to a community of practice* is an efficient means for meeting such practical and qualitative objectives. The ESCOLE+ platform dedicated to CSCL practice, evaluation, and dissemination could also help in that direction by hosting Omega+ as well as a variety of other CSCL systems.

References

Avouris, N., Komis, V., Margaritis, M., & Fidas, C. (2004). Modeling space: A tool for synchronous collaborative problem solving. In *Proceedings of AACE Conf. ED-MEDIA'04* (pp. 381–386).

Baker, M. J. (1997). Structuring and scaffolding reflective interactions in a computer-supported collaborative learning environment. Invited Symposium on Tools and Interactions in Distributed Cognitive Systems. In *Proceedings of the European Conference for Research in Learning and Instruction*, August, Athens, Greece.

Baker, M. J., & Lund, K. (1996). Flexibly structuring the interaction in a CSCL environment. In P. Brna, A. Paiva, & J. Self (Eds.), *Proceedings of the Euro AIED Conference* (pp. 401–407). Lisbon: Edições Colibri.

Baker, M. J., Quignard, M., Lund, K., & Séjourné, A. (2003). Computer-supported collaborative learning in the space of debate. In B. Wasson, S. Ludvigsen, & U. Hoppe (Eds.), *CSCL:*

Designing for change in networked learning environments, Proceedings of the International Conference on Computer Support for Collaborative Learning 2003 (pp. 11–20). Dodrecht, The Netherlands: Kluwer.

Barros, B., & Verdejo, M. (2000). Analysing student interaction processes in order to improve collaboration. The DEGREE approach, *International Journal of Artificial Intelligence in Education*, *11*, 221–241.

Benner, K., Feather, M., Johnson, W., & Zorman, L. (1993). Utilizing scenarios in the software development process. In N. Prakash, C. Rolland, & B. Pernici (Eds.), *Information system development process* (pp. 117–134). Netherlands: Elsevier.

Bødker, S., & Christiansen, E. (1997). Scenarios as springboards in design. In G. Bowker, L. Gaser, S. Star, & W. Turner (Eds.), *Social science research, technical systems and cooperative work* (pp. 217–234). Washington, District of Columbia: Aspen Institute.

Börding, J., Voss, A., Walther, J., Wolff, V., Ocakli, A., de Groot, R. et al. (2003). DUNES—Dialogic and argumentative negotiation educational software. In A. Bode, J. Desel, S. Rathmeyer, & M. Wessner (Eds.), *DeLFI 2003* (pp. 290–298). Bonn, Germany: LNI.

Bradner, E., Kellog, W. A., & Erickson, T. (1999). The adoption and use of babble: A field study of chat in the workplace. In *Proceedings of ECSCW'99* (pp. 139–158). Dodrecht, The Netherlands: Kluwer.

Clark, H., & Schaefer, E. (1989). Contributing to discourse. *Cognitive Science*, *13*, 259–294.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning and instruction: Essays in honour of Robert Glaser* (pp. 453–494). Hillsdale, New Jersey: Lawrence Erlbaum.

Constantino-González, M., & Suthers, D. (2001). Coaching collaboration in a computer-mediated learning environment. In G. Stahl (Ed.), *Computer support for collaborative learning: Foundations for a CSCL community. Proceedings of CSCL 2002* (pp. 583–586). Hillsdale, New Jersey: Lawrence Erlbaum.

Dillenbourg, P. (1999). What do you mean by collaborative learning? In P. Dillenbourg (Ed.), *Collaborative learning: Cognitive and computational approaches* (pp. 1–20). Oxford: Elsevier.

Dillenbourg, P. (2002). Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed.), *Three worlds of CSCL. Can we support CSCL?* (pp. 61–91). Heerlen: Open Universiteit Nederland.

Dillenbourg, P. (2005). Designing biases that augment socio–cognitive interactions. In R. Bromme, F. W. Hesse, & H. Spada (Eds.), *Barriers and biases in computer-mediated knowledge communication and how they may be overcome* (p. 1). Berlin Heidelberg New York: Springer.

Dimitracopoulou, A. (2005). Designing collaborative learning systems: Current trends & future research agenda. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), *Proceedings of Computer Supported Collaborative Learning 2005: The next 10 years!* (pp. 115–124). Mahwah, New Jersey: Lawrence Erlbaum.

Fadel, L., & Nazareth, A. (2004). Animated-chat, facial expression to support social sense of presence. In A. Soller, P. Jermann, M. Mühlenbrock, & A. M. Monés (Eds.), *Proceedings of the 2nd International Workshop on Designing Computational Models of Collaborative Learning Interaction* (pp. 95–99). Maceio, Brazil: http://www.cscl-research.com/Dr/ITS2004Workshop/proceedings.pdf.

Farnham, S., Chesley, H. R., McGhee, D. E., Kawal, R., & Landau, J. (2000). Structured online interactions: Improving the decision-making of small discussion groups. In *Proceedings of Computer Supported Cooperative Work 2000* (pp. 299–308). New York, New York: ACM.

Fidas, C., Komis, V., Avouris, N., & Dimitracopoulou, A. (2002). Collaborative problem solving using an open modeling environment. In G. Stahl (Ed.), *Computer support for collaborative learning: Foundations for a CSCL community. Proceedings of CSCL 2002* (pp. 654–655.). Hillsdale, New Jersey: Lawrence Erlbaum.

Garcia, A., & Jacobs, J. (1999). The eyes of the beholder: Understanding the turn talking system, in quasi-synchronous computer mediated communication. *Research on Language and Social Interaction*, *32*(4), 337–367.

Gogoulou, A., Gouli, E., Grigoriadou, M., & Samarakou, M. (2005). ACT: A web-based adaptive communication tool. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), *Proceedings of Computer Supported Collaborative Learning 2005: The next 10 years!* (pp. 180–189). Mahwah, New Jersey: Lawrence Erlbaum.

Haatainen, E., & Korhonen, K. (2002). *Guidelines for teacher training and technical and pedagogical support*: *ITCOLE teacher training and consulting model* (ITCOLE Project Deliverable D8.1, IST-2000-26249). Helsinki, Finland: University of Art and Design, Media Lab: http://www.euro-cscl.org/site/itcole/D8_1_guidelines_for_teach.pdf.

Hernandez, D., Asensio, J. I., & Dimitriadis, Y. (2004). IMS learning design support for the formalisation of collaborative learning flow patterns. *Proceedings of the Fourth IEEE International Conference on Advanced Learning Technologies (ICALT'04)* (pp. 350–354). Piscataway, New Jersey: IEEE.

Herrington, J., & Oliver, R. (1995). Critical characteristics of situated learning: Implications for the instructional design of multimedia. In J. Pearce & A. Ellis (Eds.), *Learning with technology* (pp. 253–262). Parkville, VA: University of Melbourne.

Holst, S. (2000). Evaluation of collaborative virtual learning environments: The state of the art. In F. Scheuermann (Ed.), *Campus 2000: Lernen in neuen Organisationsformen. Proceedings of GMW 2000 Fachtagung der Gesellschaft für Medien in der Wissenschaft* (pp.199–212). Innsbruck, Austria, September, 19.

Jackson, S., Stratford, S., Krajcik, J., & Soloway, E. (1996). Making dynamic modeling accessible to pre-college science students. *Interactive Learning Environments*, *4*(3), 233–257.

Jaspers, J., Erkens, G., & Kanselaar, G. (2001). COSAR: Collaborative writing of argumentative texts. In T. Okamoto, R. Hartley, Kinshuk, & J. P. Klus (Eds.), *Advanced learning technologies. Issues, achievements and challenges* (pp. 269–272). Piscataway, New Jersey: IEEE.

Jermann, P., Soller, A., & Muehlenbrock, M. (2001). From mirroring to guiding: A review of state of the art technology for supporting collaborative learning. In P. Dillenbourg, A. Eurelings, & K. Hakkarainen (Eds.), *Proceedings of Euro CSCL 2001: European Perspectives on Computer-supported Collaborative Learning* (pp. 324–331). Maastricht, The Netherlands, University of Maastricht.

Kurlander, D., Skelly, T., & Salesin, D. (1996). Comic chat. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (pp. 225–236). New York, New York: Addison-Wesley.

Lipponen, L. (2002). Exploring foundations for computer-supported collaborative learning. In G. Stahl (Ed.), *Computer support for collaborative learning: Foundations for a CSCL community. Proceedings of CSCL 2002* (pp. 72–81). Hillsdale, New Jersey: Lawrence Erlbaum.

Löhner, S., van Joolingen, W., & Savelsbergh, E. (2003). The effect of external representation on constructing computer models of complex phenomena. Instructional Science, Volume 31 (pp. 395–418). Dodrecht, The Netherlands: Kluwer.

Lonchamp, J. (2005). A structured chat framework for distributed educational settings. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), *Proceedings of computer supported collaborative learning 2005: The next 10 years!* (pp. 403–407). Mahwah, New Jersey: Lawrence Erlbaum.

McManus, M., & Aiken, R. (1996). Teaching collaborative skills with a group leader computer tutor. *Education and Information Technologies*, *1*, 75–96.

Miao, Y., Hoeksema, K., Hoppe, H. U., & Harrer, A. (2005). CSCL scripts: Modeling features and potential use. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), *Proceedings of computer supported collaborative learning 2005: The next 10 years!* (pp. 423–432). Mahwah, New Jersey: Lawrence Erlbaum.

Moore, M. G. (1993). Transactional distance theory. In D. Keegan (Ed.), *Theoretical principles of distance education* (pp. 22–38). London: Routledge.

Mühlpfordt, M., & Wessner, M. (2005). Explicit referencing in chat supports collaborative learning. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), *Proceedings of Computer Supported Collaborative Learning 2005: The next 10 years!* (pp. 460–469). Mahwah, New Jersey: Lawrence Erlbaum.

Münzer, S., & Xiao, B. (2004). Synchronous cooperative distance learning at the workplace: Technology and other factors determining the quality of the learning process. In K. Tochterman & H. Maurer (Eds.), *Proceedings of I-Know '04* (pp. 543–550). Graz, Austria: Know-Center Austria.

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

O'Neil, J., & Martin, D. (2003). Text chat in action. In *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work* (pp. 40–49). New York, New York: ACM.

Pea, R., Edelson, D., & Gomez, L. (1994). Distributed collaborative science learning using scientific visualization and wideband telecommunications. Symposium Multimedia information systems for science and engineering education: Harnessing technologies. Symposium conducted at the 160th Meeting of the American Association for the Advancement of Science, San Francisco, California, February.

Pfister, H.-R., & Mühlpfordt, M. (2002). Supporting discourse in a synchronous learning environment: The learning protocol approach. In G. Stahl (Ed.), *Computer support for collaborative learning: Foundations for a CSCL community: Proceedings of CSCL 2002* (pp. 581–589).

Pimentel, M., Fuks, H., & Lucena, C. (2005). Mediated chat development process: Avoiding chat confusion on educational debates. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), *Proceedings of Computer Supported Collaborative Learning 2005: The next 10 years!* (pp. 494–498). Mahwah, New Jersey: Lawrence Erlbaum.

Pinkwart, N. (2003). A plug-in architecture for graph based collaborative modeling systems. In U. Hoppe, F. Verdejo, & J. Kay (Eds.), *Shaping the future of learning through intelligent technologies: Proceedings of the 11th Conference on Artificial Intelligence in Education* (pp. 535–536). Amsterdam, The Netherlands: IOS.

Robertson, J., Good, J., & Pain, H. (1998). Better Blether: The design and evaluation of a discussion tool for education. *International Journal of Artificial Intelligence in Education*, *9*, 219–236.

Rosson, M., & Carroll, J. (2002). *Usability engineering: Scenario-based development of human–computer interaction*. San Francisco, California: Morgan Kaufmann.

Scardamalia, M. (2003). Knowledge Forum (Advances beyond CSILE). *Journal of Distance Education*, *17*(Suppl. 3), 23–28.

Siebra, S., Christ, C., Queiroz, A., Tedesco, P., & Barros, F. (2004). SmartChat—An intelligent environment for collaborative discussions. In J. C. Lester et al. (Eds.), *LNCS 3220:Proceedings of ITS 2004* (pp. 315–324). Berlin Heidelberg New York: Springer.

Singley, M., Singh, M., Fairweather, P., Farrell, R., & Swerling, S. (2000). Algebra jam: supporting teamwork and managing roles in a collaborative learning environment. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work (CSCW)* (pp. 145–154). New York, New York: ACM.

Smith, M., Cadiz, J. J., & Burkhalter, B. (2000). Conversation trees and threaded chats. In *Proceedings of CSCW 2000* (pp. 97–105). New York, New York: ACM.

Soller, A., Linton, F., Goodman, B., & Lesgold, A. (1999). Toward intelligent analysis and support of collaborative learning interaction. In *Proceedings of the ninth International Conference on Artificial Intelligence in Education* (pp. 75–82). Amsterdam, The Netherlands: IOS.

Spector, J. M. (2000). System dynamics and interactive learning environments: Lessons learned and implications for the future. *Simulation & Gaming*, *31*(4), 528–535.

Stahl, G. (2004). Groupware goes to school: adapting BSCW to the classroom. *International Journal of Computer Applications in Technology*, *19*(3–4), 162–174.

Suchman, L. (1987). *Plans and situated action*. Cambridge, Massachusetts: Cambridge University Press.

Suthers, D. (2005). Technology affordances for intersubjective learning: A thematic agenda for CSCL. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), *Proceedings of Computer Supported Collaborative Learning 2005: The next 10 years!* (pp. 662–671). Mahwah, New Jersey: Lawrence Erlbaum.

Suthers, D., & Jones, D. (1997). An architecture for intelligent collaborative educational systems. In *Proceedings of 8th World Conference on Artificial Intelligence in Education (AI-ED'97)* (pp. 55–62). Amsterdam, The Netherlands: IOS.

Tattersall, C., Vogten, H., & Hermans, H. (2005). The edubox learning design player. In Koper & Tattersall (Eds.), *Learning design* (pp. 303–310). Berlin Heidelberg New York: Springer.

van Joolingen, W., de Jong, T., Lazonder, A., Savelsbergh, E., & Manlove, S. (2005). Co-Lab: research and development of an online learning environment for collaborative scientific discovery learning. *Computers in Human Behavior*, *21*, 671–688.

Viegas, F. B., & Donath, J. S. (1999). Chat circles. In *Proceedings of CHI 1999* (pp. 9–16). New York, New York: ACM.

Vieira, A., Teixeira, L., Timoteo, A., Tedesco, P., & Barros, F. (2004). Analyzing online collaborative dialogues: The OXEnTCHE-chat. In J. C. Lester et al. (Eds.), *LNCS 3220: Proceedings of ITS 2004* (pp. 315–324). Berlin Heidelberg New York: Springer.

von Krogh, G., Spaeth, S., & Lakhani, K. (2003). Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, *32*, 1217–1241.

Vronay, D., Smith, M., & Drucker, S. (1999). Streaming media interfaces for chat. In *Proceedings of UIST 1999* (pp. 19–26). New York, New York: ACM.

Vygotsky, L. (1978). *Mind in Society: The development of higher psychological processes*. Cambridge, Massachusetts: Harvard University Press.

Whitehead, R. K., & Stotts, D. (2000). *ProChat: Dynamic formal collaboration protocols in a chat tool for handled collaboration* (Technical Report UNC TR00-016). Durham, University of North Carolina.

Winograd, T., & Flores, F. (1986). *Understanding computers and cognition*. New Jersey: Ablex.

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, *17*, 89–100.

Zhang, J. (1997). The nature of external representations in problem solving. *Cognitive Science*, *21*(2), 179–217.