Research and Practice in Technology Enhanced Learning
Vol. 5, No. 3 (2010) 185–203

DOI: 10.1142/S1793206810000888

PROCESS SUPPORT FOR COLLABORATIVE INQUIRY LEARNING

NIELS PINKWART
Clausthal University of Technology, Department of Informatics
Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany
niels.pinkwart@tu-clausthal.de
http://hcis.in.tu-clausthal.de

ANDREAS HARRER
Catholic University Eichstätt-Ingolstadt
Department of Computer Science
Ostenstr. 14, 85072 Eichstätt, Germany
andreas.harrer@ku-eichstaett.de

MARKUS KUHN
University of Duisburg-Essen
Department of Computer Science and Applied Cognitive Science
Lotharstr. 63/65, 47048 Duisburg, Germany
kuhn@collide.info

In science, the term model refers to a schematic, simplified and idealized representation of an object or a domain, in which the relations and functions of the elements are made explicit. Modeling is understood as the activity of creating, manipulating and using models. This article presents a discussion of how collaborative modeling activities, an essential part of collaborative inquiry learning, can be facilitated by supporting group processes either by embedding concrete learning tools or through externalized learning process specifications that can provide scaffolding or feedback. We illustrate these design options with a small-scale classroom study where a collaborative modeling environment called Cool Modes was employed to support students as they investigated topics of probability. The study involved multiple learner groups who first worked on a task individually or in small groups, and then pooled their results as part of an inquiry cycle.

Keywords: Inquiry learning; collaborative modeling; process support.

1. The Role of Collaborative Modeling in Inquiry Learning

In science, the term *model* refers to a schematic, simplified and idealized representation of an object or a domain, in which the relations and functions of the elements are made explicit. There is an analogy between the model and the object it describes in the sense that these two have similar structures (attributes and relations between these attributes).

Models are generally expressed in notations or visualizations which emphasize the specific attributes and relations that the model makes explicit. These notation systems are called modeling languages. Some modeling languages like Petri nets (Petri, 1962) or System Dynamics (Forrester, 1968) describe models that can be executed or serve as input for executable simulations.

Modeling is understood as the activity of creating, manipulating and using models. Obviously, computers can support modeling activities in that they can serve as tools that interpret models and execute model-based simulations. The idea of using computers as active tools for modeling is indeed not new: Kay and Goldberg (1977/2001) described their vision of the Dynabook — a notebook size "self-contained knowledge manipulator" that can serve as an integrated repository for dynamic media and simulations. With technology as available today, such a use of computers for modeling and simulation is established. With some right, these functions could be called "basic support for modeling" nowadays.

1.1. *Modeling as an educational activity*

A specific field where modeling can be beneficial is education. As models are a simplified and manageable means of understanding complex real-world phenomena, the importance of modeling in science education is evident: "Modeling is a central skill in scientific reasoning and provides a way of articulating knowledge. Learning to formulate, test, and revise models is a crucial aspect of understanding science and is critical to helping students become active, lifelong learners" (Bredeweg & Forbus, 2003, p. 35).

Already the hypothetical device of Kay and Goldberg offered the option to run simulations and to interact with them. Their vision included educational usage: "Mathematics could become a living language in which children could cause exciting things to happen. Laboratory experiments and simulations too expensive or difficult to prepare could easily be demonstrated" (p. 177).

From today's perspective, these points seem to be fairly general. But also beyond them, some pedagogical motivations for including computers in educational modeling activities in a way that exceeds the pure "model execution" functionality have been proposed (Jonassen, 2000; Perkins, 1991). These include the Microworlds idea of Papert (1980) in which computer-based tools would provide children with a whole range of transformative developmental experiences, the work of Wild (1996) who argued that, as computer-based modeling externalizes thinking and knowledge, models become tools for the conscious manipulation of thought, and the position of Kurtz dos Santos and Ogborn (1994) who state that the degree of flexibility and interactivity that computers offer (e.g. parameters of models can be changed quickly, and it is easy to compare model variations to each other) enhances learner's understanding of problems. Also the function of computers as a means to explore different visual representations of models can be beneficial for learning (Ainsworth, 1999).

However, modeling in the sense of creating and manipulating (simplified) representations that explain complex real world phenomena is a challenging task for students. As Sierhuis and Selvin (1996) suggest, collaboration can be helpful here since it may reduce the complexity of the modeling task. While the work of Sierhuis and Selvin is not restricted to educational settings (but addresses modeling in a more general sense), research has also acknowledged the potential of using modeling as a collaborative learning technique (Hoppe, 2004; van Joolingen, 2000). Hoppe (2004) builds upon the concept of "mind tools" as proposed by Jonassen (2000) — computer-based tools and learning environments are intellectual partners of the learner in order to engage and facilitate critical thinking and higher order learning — and introduces the notion of collaborative mind tools as environments that inherently synthesize communication and collaboration support with interactive and constructive features, resulting in "computational objects to jointly think with" (Hoppe, 2004). Hoppe argues that the educational benefits of peer collaboration (such as knowledge exchange), combined with the learning gains achievable through "constructive tools" (Jonassen *et al.*, 1999) such as modeling tools, can be very fruitful. This general position is supported also from a variety of educational approaches such as Model Facilitated Learning (Milrad *et al.*, 2002) or Scientific Discovery Learning (De Jong & van Joolingen, 1998; van Joolingen, 2000). Also for the specific viewpoint of Inquiry Learning (White & Frederiksen, 2000), the argument can be made that modeling has a key function in the inquiry cycle and can be supported by means of collaboration (cf. Tsovaltzi *et al.*, 2010).

1.2. *Technology based modeling support*

Modern networked digital technology can, through archival and retrieval functions, foster the exchange and reuse of modeling material. Clearly, today's technology principally enables a cooperative use of modeling tools. Yet, the question of the specific kinds of appropriate support that computers can offer for collaborative modeling tasks, particularly in education, is not completely answered and an issue of current research. Here, contributions have been made — among others — by van Joolingen and Löhner (2001) and Or-Bach (2003). The former paper focuses on the use of different representations (textual, graphical or output oriented) for collaborative modeling tasks in education, the latter reflects about design decisions in terms of interaction modes and support mechanisms for computer-based modeling tools for learning purposes. Addressing the issue of computer technology for modeling, it is not surprising that both papers describe the function of the computer as an active medium that can be used to run simulations generated from the constructed models. This level of support can be classified as *task-specific* support which employs the computer as a "simulation machine" for models. Indeed, a second joint position of the papers is that they see the basic approach for supporting collaborative modeling in *process* support, namely in supporting the learners in going through the process of jointly working on the model and co-constructing it. Here, the computer

has the additional role of a communication medium. This technique of sharing of external representations with the option of communicating through the constructed artefact can frequently be observed also in existing collaborative modeling tools like Belvedere (Suthers *et al.*, 1995), ModellingSpace (Margaritis *et al.*, 2003), Co-Lab (van Joolingen *et al.*, 2005), or Digalo (Schwarz & Glassner, 2007). The desire to adequately support flexible interactions with both the collaboration partners and the jointly manipulated artefact leads to a number of design challenges.

In the remainder of this article, we first discuss these design challenges and then subsequently describe the design of two technological solutions to support students in collaborative modeling activities. One of these solutions is on the micro level (tool internal) and addresses forms of flexibly offering collaboration support in shared workspace systems; the second one is on the macro level (tool external) and relies on externalized learning process scripts and engines which are used as "scripted remote controls" for educational modeling tools. Furthermore, we present a small *in situ* classroom study in which one of the design solutions was employed to provide technological support to a group of learners in the domain of probability.

2. Design Challenges

Two design principles seem to be key factors for a successful collaborative modeling tool. First, flexibility with respect to the supported representations is a crucial point. This includes the support of multiple representations of problems, different phases in the activity (which correspond to different representations), and dynamic interactive changes of simulation parameters (Löhner *et al.*, 2003; Ainsworth, 1999; Kurtz dos Santos & Ogborn, 1994). The second factor is interoperability between different models, different model representations and different modes of collaboration, to enable mixed structures and the flexible creation of customized expressions with "educational building blocks" (Löhner *et al.*, 2003; Roschelle *et al.*, 1999). In particular with respect to the second point, several challenges can be distinguished:

- *Syntactic* interoperability. The modeling tool should allow the use of mixed external representations — heterogeneous (or "hybrid") models that are composed of elements which belong to different modeling languages.
- *Semantic* interoperability. Going beyond the ability to allow for mixed representations, the tool should provide interfaces that facilitate model integration issues. A design goal is to allow tools to reuse elements defined within others, share semantic definitions and also offer support for easy transformations of expressions from one modeling language to another (Heiler, 1995; Read *et al.*, 2003).
- *Social* interoperability. Given the targeted educational use, the tool must be easy to operate. It should allow for an integrated collaborative use in a way that enables co-learners to remain in their social work context, even when switching to other means of expression. For example, when a learning group has just discussed different hypotheses (using a brainstorming tool) and then opens a simulation tool

to test the hypotheses, social interoperability means that the group work context is still maintained in the new tool — e.g. by being able to conduct the simulation jointly, or simply by being aware of the actions of the peer learners.

- *Process* interoperability. The dimensions of process interoperability that a tool can support are the avoidance of media breaks (which a change from one software tool to another is likely to cause), a flexible collaboration support to allow for task-compliant collaborative settings, and the support for work phases that go in line with the needs of different modeling tasks. For example, when a group decides to collaborate in four steps — first working together as a whole, then splitting up into subgroups, then continuing individually, and finally working in the plenum — then process interoperability can be characterized through a seamless flow of phase results, in the best case irrespective of which tools were used in the phases.

These four interoperability challenges refer to different aspects of tool support for collaborative modeling. Yet, they are not fully independent of each other: semantic interoperability requires syntactic interoperability as a prerequisite, and social interoperability is usually connected with process interoperability (since especially longer-term collaboration is a social process).

Existing collaborative modeling tools such as Belvedere (Suthers *et al.*, 1995), Co-Lab (van Joolingen *et al.*, 2005), ModellingSpace (Margaritis *et al.*, 2003), Digalo (Schwarz & Glassner, 2007) or Collect-UML (Baghaei *et al.*, 2007) do not fully meet these challenges due to a lack of support for multiple interoperable modeling languages and/or limitations in terms of process interoperability.

The remainder of this paper focuses primarily on the aspects of process and social interoperability, since these are the most specific factors in a collaborative learning context.

3. The Micro Level: New Forms for Shared Workspaces

Cool Modes (COllaborative Open Learning and Modeling System) is an educationally oriented groupware tool that allows co-constructive modeling activities with graph-based models. A distinctive feature of Cool Modes is its flexibility in terms of supporting collaborative settings via highly configurable "shared workspaces" technologies. In the following, we first briefly describe these collaboration features of the system, and then illustrate them with a case study — a course sequence in the domain of probability that was conducted using Cool Modes.

3.1. *Technical features of Cool Modes*

Essentially, the Cool Modes system relies on two foundations. First, it supports visual graph-based modeling languages (called "Reference Frames") to be externally defined and "plugged into" the system at runtime. Unless explicitly restricted, elements from these languages can be used together to build heterogeneous models.

The second fundamental concept in Cool Modes is the use of workspaces — these are the places for the creation and direct manipulation of the graph-based models. Cool Modes supports multiple workspaces, which can be arranged freely on the desktop that the system offers. Similar to a number of other environments (e.g. Suthers *et al.*, 1995; Margaritis *et al.*, 2003; van Joolingen *et al.*, 2005), the core collaboration support in Cool Modes relies on the principle of sharing workspaces. Changes caused by a user in a shared workspace are propagated to the corresponding workspaces in the applications of the collaboration partners. This leads to the conceptual group interface of a "shared graph space" with a WYSIWIS (What You See Is What I See) usage metaphor. Multiple workspaces per desktop are possible, allowing students to maintain private workspaces together with shared workspaces on the screen, and thus "publish" results to the group using simple drag and drop operations, or to develop private "try-out" models while watching what the rest of the group is doing in the shared space.

One type of user support provided by Cool Modes is based on the observation that modeling activities typically involve different phases, which may correspond to different representations, different tools, or different usage modes like exploration or simulation (Löhner *et al.*, 2003; Ainsworth, 1999; Kurtz dos Santos & Ogborn, 1994). Oftentimes, phases in a collaborative modeling task can be associated to specific modeling languages and tools (e.g. exploratory design with causal feedback loop models, formal model construction with System Dynamics models, and argumentation with concept maps). To support the collaboration process also when users switch between tools, Cool Modes allows learners to notify their peers when they use a new modeling language (i.e. a new Reference Frame). The co-learners can then choose to use the new modeling language as well. This is a way of maintaining a common group phase, since all collaboration partners are made aware about the "change of language" and the corresponding "change of activity phase".

Modeling is an activity which contains both constructive phases in which models are built (or revised), and testing phases in which the created models are tested and "run" in simulations. Therefore, apart from phases that are directly related to specific modeling languages, it is also reasonable to distinguish between phases based on usage modes (distinguishing between "editing" and "running" models). Here, collaborative modeling situations have specific requirements: if one user intends to edit a model, while concurrently another one tries to conduct test runs on it, this may lead to confusion and unintended inconsistency. To offer a means of coordination here, the Cool Modes environment allows the selection of one of three interaction modes. The following modes are supported:

- An *integrated* mode, which allows both editing of model graphs and simulations of models (i.e. one user could change a model that another user is simulating at the same time).
- An *edit* mode, which lets the users co-construct and revise models, but does not support any simulations.

- A *simulation* mode in which the model structure is fixed and simulations can be conducted (i.e. edits are impossible).

Similar to the propagation of Reference Frame changes, the interaction modes support the collaborators in maintaining a shared understanding of the current state of the group activity.

In addition, also the artefact being shared can be flexibly selected in Cool Modes (cf. Table 1). This broadens the application areas of the tool and enables a variety of collaboration scenarios, such as personal annotations in a shared workspace, or "try-out" private test versions of models embedded into collaboratively used workspaces. As stated above, the classical shared workspace metaphor already suggests mixtures of private and group workspaces. The Cool Modes mechanisms for workspace sharing go far beyond this "shared or private" all-or-nothing choice. They have been designed based on the premise not to restrict the educational designer in the collaborative settings he can orchestrate with the tool. However, arbitrary flexibility can cause undesired inconsistencies of model semantics: e.g.

Table 1. The Cool Modes synchronization modes.

Synchronization Mode	Description	Example Use Case
Application	The complete Cool Modes application (i.e. all workspaces), is synchronized. Private actions are not possible within the system. Instead, each user has a complete view on what the others are currently doing.	"Traditional" group work in a shared workspace environment.
Workspace	Private workspaces and shared workspaces are offered synchronously.	Users who work on "test models" and publish them to the group after having verified them.
Workspace Layer	A workspace can be partially synchronized in the sense that some "layers" are shared, while others are private.	Users can make individual handwritten annotations attached to jointly used models.
Reference Frame	Model elements that belong to user-selected Reference Frames (i.e. modeling languages) are automatically "published" in the workspace.	This mode allows a pre-selection of private and shared entities, for instance, to a priori determine that "model elements" are shared, while "comment elements" are kept private.
Synchronization Context	This "finest" level of synchronization allows the user to specify exactly the model elements he wants to share. These are shared together with their "synchronization context", which is a set of other model elements that ensure a consistent semantics.	Here, a user can make private additions to a shared model as long as they do not have an impact on the semantics of the shared model. If they do, the user has the choice either to add the private elements to the shared model, or to work on a private copy of the overall model.

if only single model elements are shared, the local model semantics can vary unpredictably between the applications of the collaboration peers, making successful group work impossible (Pinkwart, 2005). As a result of the trade-offs between flexibility and consistency, Cool Modes offers the five synchronization modes shown in Table 1, which are characterized through their atomic element of synchronization.

In the following, we present an extended example that demonstrates the educational usefulness of partial sharing mechanism in a classroom situation.

3.2. *A case study: Learning probability with Cool Modes*

Learning probability is a relatively modern aspect of school education and a suitable field for inquiry learning. Studies of Armstrong (1972) and Fischbein (1975) show that hands-on experiments are necessary to enable students to understand fundamental probability concepts concerning stochastics experiments and frequencies. Fischbein (1975) points out that practical experience with probabilities provides an ideal way of familiarizing children with the fundamental concepts of science, such as prediction, experiment and verification. In school, exploratory learning is often practiced by throwing dice or coins and examining the outcomes of such experiments. Students are often motivated to find out more about "chance", especially concerning gambles.

Hands-on experiments in probability however soon reach their limits. The continuous repetition of similar experiments, which is often needed to come to meaningful results, is both tiring for the students and time-consuming. If one is sceptical about technical solutions to replace hands-on experiments with computer simulations (which is a reasonable standpoint for educators), one can argue that starting with manual experiments, then demonstrating that technology-based simulations yield the same results, and finally running simulations to investigate effects "in the large" is an acceptable strategy which fulfils the needs both for authenticity and for many repetitions of experiments. Since the 1990s, several software tools like Prob Sim (Konold, 1995) were developed offering students "pre-built microworlds" and supporting them in modeling, simulating and analyzing probabilistic experiments. Studies based on the use of the Probability Inquiry Environment (Enyedy, 2000) which arranges collaborative learning processes in form of inquiry cycles indicate that students can acquire a deeper understanding of concepts like fairness and the law of great numbers as a result of the use of computer-based tools. Such experiences show that learning probability and modeling, both relatively modern aspects of mathematics education, play an essential role to initiate learning processes, and can be supported by adequate technology.

To illustrate how collaborative software tools can enrich instruction through social and process interoperability, we conducted an *in situ* study which took place with a school class (approx. 25 students) of 9th graders at the Elsa-Brändström-Gymnasium in Oberhausen, Germany. The study, which comprised nine lessons of

45 minutes each, was part of the regular classes of the students. It was taught by the mathematics teacher of this class who took notes after the lessons and reported on his experiences and impressions in a semi-structured interview.

The technological setup for the study consisted of a computer room in the school that was equipped with an interactive whiteboard and several computers with tablet devices allowing for pen-based input. Students could do the modeling tasks in Cool Modes and, using the tablets, were also able to add hand-written annotations to the model in Cool Modes, naturally acting with the pen while building the model in the workspace (cf. Figure 1). The tasks at the computer were solved in small groups of 2–4 students sharing one device.

The lesson sequence (and, thus, the tasks the students had to solve) dealt with probability and comprised five main steps (A to E). The main topic was the birthday paradox, an ideal problem to leave the normal ways of math instruction and to initiate inquiry learning. Beginning with easy "urn model experiments" such as "throwing" one or two dice or drawing colored balls, the students first had a chance to get familiar with the software in the first lessons (Step A). At the same time they could freshen up some basic knowledge, including the conditions for adding or multiplying probabilities in Laplace experiments.

The birthday paradox "How high is the probability that in a group of n people at least two have a common birthday?" was then introduced through a betting contest (following the crucial steps described by Konold (1991), who suggested to test own beliefs first against the beliefs of others, then against own beliefs about other related

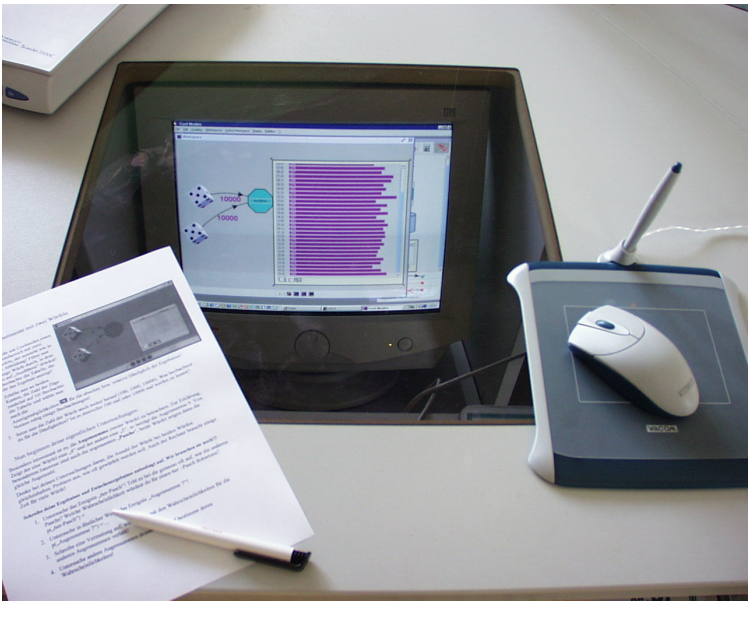

Figure 1. The stochastics tool.

things, and finally against empirical evidence) to increase the motivation of the students. For students it is not evident how to calculate the probability directly using the complementary event "no common birthday". It was intended that they use the modeling environment to develop an experimental solution first to later prepare a theoretical, algebraic solution.

The students had to build an adequate model in Cool Modes to explore the problem (Step B). Here, the Cool Modes software and the classroom hardware equipment allowed several forms of construction with the whole class — e.g. modeling in one shared workspace connecting all computers, modeling in a single jointly used workspace using the interactive whiteboard followed by sharing of the model, or reconstruction by the students. The teacher decided to have students co-construct the model (all learners together with the teacher, who moderated the process) with subsequent reconstruction of the model at the students' workplaces to prepare self-regulated modeling for later course units. The Cool Modes stochastics environment provides a calendar urn, i.e. a representation of 365 days (see left element in Figure 2) which was supposed to be used for this task.

The students then had to perform at least 10 experiments to determine how often the event "at least one common birthday" occurs with a group with 24 members (Step C). The experimental work was arranged in small groups working in one environment or in synchronously used shared environments. For the sharing of outcomes between groups, the partial workspace coupling functions of Cool Modes (as described above) have been employed. Specifically, a common shared object in form of a table for group wise reporting of experimentation results was employed

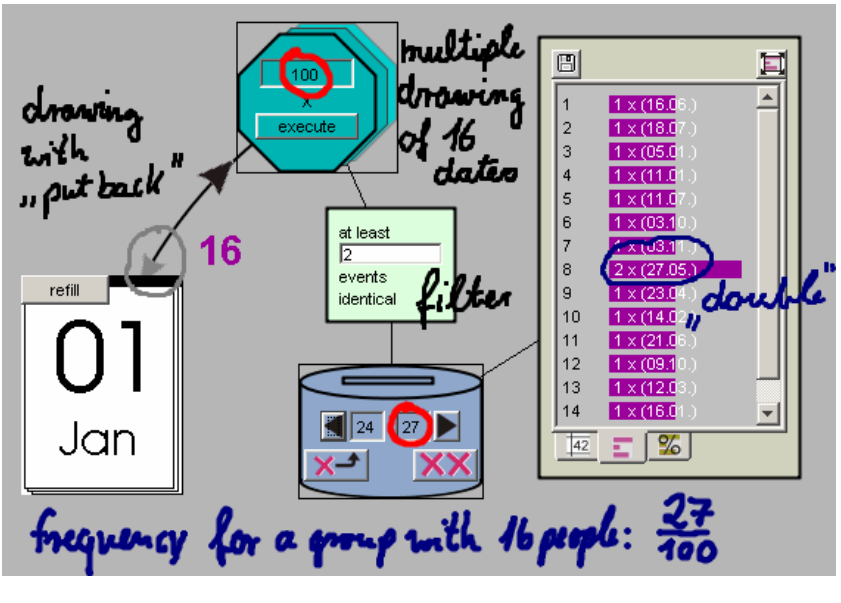

Figure 2. Shared model for the birthday experiment.

to prepare the computation of the relative frequency over all experiments done by the class.

A varied task then started a new process of inquiry. In this second scenario, the students had to modify the previously created "birthday microworld" to explore the group size for which the probability for "at least one common birthday" is 50 percent (Step D). Now, based on an implicit or explicit hypothesis, they had to decrease or increase the group size before analyzing and counting positive outcomes of experiments. This time the students reported their results permanently via a shared table (again, using partial workspace synchronization). So each group could make use of all reported outcomes and alter its procedure based on peer feedback.

In the final lesson, the empirical results were compared with the correct probability which was determined algebraically using the complementary event "no common birthday" (Step E).

During the whole course the teacher constantly made use of the interactive whiteboard to structure and document the class' work and their results. These documents were archived and sent to the students via email.

Overall, the results of the study were positive and encouraging, in particular compared to "traditional" lessons without computer support. In a semi-structured interview, this was stated by the class teacher who has a lot of experience in teaching probability and whose "traditional" lessons also involve experiments and result discussion, but no computer support.

In simulating and analyzing experiments, students were able to build up probabilistic concepts based on own, empirically grounded experiences. Even complex problems, based on urn experiments and automatic analysis, were collaboratively modeled, simulated and examined using Cool Modes with a corresponding modeling language and the flexible workspace sharing mechanisms. Figure 2, which contains the results of student's work in Step B of the study, illustrates this. Students had no difficulty co-constructing a basic probability model quickly using the "calendar urn" element, drawing one date after the other 16 times to simulate a given group size of 16 people (top element in Figure 2, the 100 indicating the number of simulation runs). The model was then supplemented by a filter, counting "same birthdays" (middle) and a bar diagram to visualize multiple birthdays (right). The success in the immediate construction of a suitable model by the student group at the very beginning of the lesson and the student's success in individually reconstructing the jointly created model indicate that the experimentation tools (e.g. random devices and the modes of drawing) were suitable for the tasks.

Furthermore, students reported to the teacher that they liked collaborating using Cool Modes in the forms offered in the lesson sequence. During the lessons, the teacher observed a high number of students who showed greater motivation than usual. Some did not even pay attention to end of lessons or breaks in between. Further, students asked to use the software also beyond the instruction purposes. The adequateness of the used representation (i.e. the stochastics tools) that was observed in the described introductory lesson was confirmed in a class test, in which

almost all students proved their competence in constructing a model in the stochastics modeling language which they were unfamiliar with before the study.

Collaboration was used successfully in the phases of building hypotheses and constructing models. The teacher noticed that during the phases of simulation and analysis, students were able to report about their work, but had difficulties to utilize results reported by other groups. We hypothesized that the collaborative setting of Step D (result table sharing) would influence the strategies of the groups or the class itself — yet, it turned out that the students had great difficulties to make use of the additional information. Just one out of seven groups actually used results of others to find a proper solution. This indicates that students might either need to be trained in being aware of others' work to enrich their own exploration and data and thus benefit from collaboration, or that explicit learning phases designed for inter-group communication should be foreseen.

Compared to traditional lessons without computer support, the teacher observed improvement in the field of social interoperability caused by the software environment: the use of Cool Modes facilitated the information flow between learners during the whole learning and inquiry process beginning with co-constructive work phases (designing experiments and comparing results) and ending up in the presentation of findings (using the interactive whiteboard as a "forum" for class discussions). This would not have been possible as seamlessly in traditional settings where the same information flow would have required a lot of manual and time-consuming copying and data collecting activities. In that sense, Cool Modes served as a suitable tool to support and accompany the collaborative inquiry process in the classroom study by facilitating collaboration and allowing for collaborative modeling activities.

4. The Macro Level: Externalizing Learning Scripts

The concrete collaborative classroom scenarios designed and implemented by the teacher have a much greater potential for usage than just to be used once in the conducted lesson: they can also be considered as templates of learning design that may be re-used for further different modeling activities with the same or another class, or even in other contexts. This can be done by using roles of actors (e.g. pupil, group coordinator, presenter or teacher) instead of specific persons.

Additionally a teacher usually has specific *sequences* of learning activities in mind and follows this plan during the lesson. Yet only few learning environments provide support to the teacher in planning and conducting series of collaborative learning activities. Co-Lab (van Joolingen *et al.*, 2005) has an implicit process model that manifests itself in the process coordinator, a tool that gives orientation to the student about the process steps he has already performed. Making these process models explicit opens the potential to share the pedagogical sequence and rationale with colleagues, but also to make these models executable in computer systems such that the process is automatically sequenced as planned by the teacher.

As described, our collaborative application Cool Modes is very versatile due to the wide variety of available modeling languages and different social arrangements that are feasible with it, but on the other hand setting up the scenario and having a pedagogically reasonable sequence of learning activities has to be done manually by the practitioner and without explicit process structure within the system.

The explicit definition of learning designs attracted a lot of interest in the last years and resulted in proposals of educational modeling languages, such as EML and its successor IMS/LD (IMS, 2007). The definition of learning designs for collaborative scenarios tends to be much more complex than the designs for individual learning, because different group situations and roles therein have to be specified. Related work, such as Hernandez *et al.* (2004) showed that some aspects of complex collaborative designs (also called Collaborative Learning Patterns) are not represented properly in IMS/LD and extensions for the collaborative aspects are necessary. Miao *et al.* (2005) discussed the appropriateness of IMS/LD with respect to concepts like explicit representation of groups or artefacts produced during the collaboration, and came to the conclusion that these concepts might be emulated by IMS/LD, i.e. could be mapped from the desired concepts to less abstract LD constructs.

An especially attractive aspect about the IMS/LD standard for our context is the availability of learning design engines (LDE), such as CopperCore, that are able to "play" a learning design, i.e. automatically sequence learning activities specified by the designer.

To achieve social and process interoperability for Cool Modes also at the level of coarse-grained learning activities, we currently work on the integration of our learning support environment with a learning design engine without compromising either of the two sides on the implementation level. An important benefit of such an approach is that the implementation of a process model from scratch for Cool Modes is not necessary, if we can meet the challenge of integrating the pre-existing systems in a flexible, interoperable architecture.

Harrer *et al.* (2005) present such an architecture that aims at a clear separation between the learning design engine, the specification and implementation of the learning flow in the representation of an LD document, and the learning support environments (LSEs). They assumed that the learners interact exclusively with the LSE, in our case Cool Modes, without having to know anything about being "scripted" or "scaffolded" by the LDE or the LD document, respectively. According to Vogten *et al.* (2005), learning design engines can be considered as a collection of finite state machines that react to the change of a learning situation by sending events. In the loosely-coupled connection of an engine with a learning support environment presented in the figure below, the engine controls the learning environment with output events (such as "start a new phase", event 1 in Figure 3), that are interpreted by the environment with its existing functionality (such as "create new workspace", which changes the configuration of the LSE with event 1.1). The learners interacting with the learning support environment also create

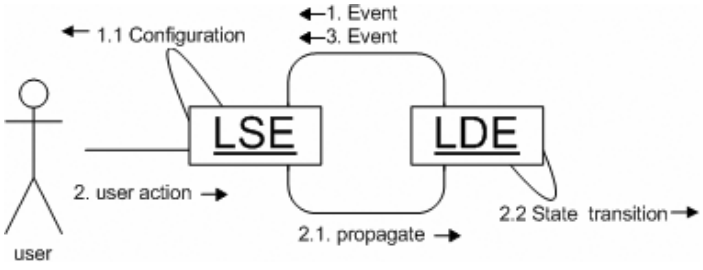

Figure 3. Regulation cycle.

events (user action 2), such as "phase is completed" (either directly or monitored by the LSE), that is propagated to the LDE (message 2.1). The situation change (message 2.2) causes the learning process to advance and will again trigger control messages (event 3) to be sent to the LSE. In that way we obtain the regulation cycle of Figure 3 with the LDE and the LSE influencing each other's behaviour.

A prototypical implementation of this approach has been conducted with Cool Modes as the learning support environment and CopperCore as the learning design engine. As an example scenario we designed a simple learning flow: First the students explore a phenomenon (e.g. multiple dice throws) and describe what they see. Then they model the observed phenomenon within a modeling environment. Every time a student states that the model is sufficient, it is "frozen" and each student has to vote if he approves the model. If there is consensus, the students present their model to the teacher. If they do not agree, the modeling activity is continued.

This scenario has been represented in IMS/LD and is usable with our proposed architecture. The learner interacts exclusively with the Cool Modes application as he is used to, but is supported by a "script" behind the scenes that is executed within the LD engine. Thus each time a phase of the learning process is finished the script defines with which activity the learners should proceed. Figure 4 shows the Cool Modes learning environment before (left) and after (right) the initiation of the voting phase by the CopperCore Engine. In the right part the voting tool was added (small icon in top right corner) and an additional window appeared to conduct the voting.

Our prototypical implementation and scenario indicate that by re-using existing systems, learning scripts can be introduced into collaborative applications without changing the applications substantially.

Admittedly, there is a general trade-off between self-regulated learning and scaffolds or guidance achieved through learning scripts. Our position is that using a learning script approach together with inquiry processes has a big potential to support students in the learning process and to support the teacher in organizing the activity: the teacher is unburdened from some micro management tasks, but can still draw pedagogical decisions that have effects on learning. Typically a teacher has some inquiry process in mind, which the students might or should follow in the

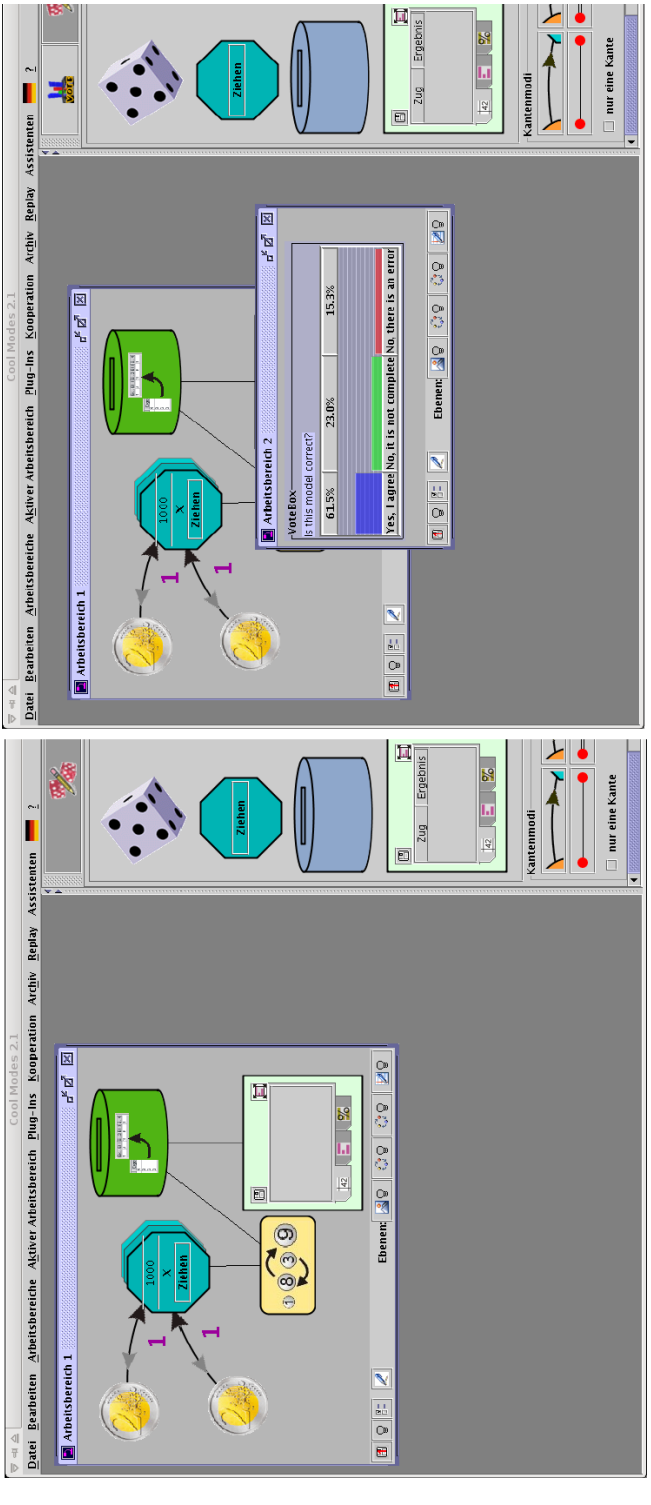

Figure 4. Cool Modes voting in LD.

Table 2. Inquiry phases and corresponding Cool Modes functions.

Inquiry Phase	Cool Modes Function
Modeling, Prediction	System Dynamics, Calculation Networks, Automata, Petri Nets
Analysis/Interpretation, Conclusion/Evaluation	Graph Plotter, Handwritten Annotations
Orientation/Asking questions, Planning	Question-Options-Criteria method, Archive Upload
Hypothesis Generation	Concept Mapping, Causal Feedback Loops
Communication	Argumentation Graph, Voting, Chat

learning scenarios. The recommended sequence of inquiry phases can be specified explicitly in a learning design description, with the additional possibility of fading out the strict process control for more experienced learners. This technology does of course not render the teacher obsolete, but in contrast supports him and puts him in a better position to help his students.

Cool Modes, as a versatile learning support environment, provides a wide variety of tools that are especially suited for different phases of the inquiry process. Some examples are shown in Table 2.

Therefore, we believe that substantial parts of an inquiry learning process can be enabled by the Cool Modes system, when the different collaboration modes are used in combination with our macro level approach. An important aspect of this approach is that it also works beyond Cool Modes — in principle, the "remote control" we propose is capable of steering, and thus integrating, multiple VLEs with their specific functions.

As a consequence, the proper tools and functionality for inquiry phases are offered at the time the teacher thinks it is appropriate, either automatically triggered by the system or interactively by the teacher. A recommended learning sequence definition could consist of using model construction with System Dynamics, experimentation with that model and against real data provided by the teacher, then evaluating the data with annotations, creation of a hypothesis with concept mapping, and finally presenting it using the archive function. Exemplifying what we have called "process interoperability" in the beginning of this paper, the whole sequence can be supported by the system in our approach, whereas previously the teacher had to take care of this process organization mainly on his own.

5. Perspectives

The technological approaches and solutions discussed in this article can be seen as advanced "enabling technologies" which open new opportunities for orchestrating and implementing computer-supported collaborative modeling activities in science education. The probability example clearly points out that designing for flexibility — a well-known principle in computer science and systems engineering — can indeed allow the teachers to implement pedagogically interesting inquiry learning

scenarios based on his educational considerations (instead of technical constraints about what is possible and what is not).

However, the added flexibility for designing collaboration tasks as discussed in this article poses new questions: modeling, in particular quantitative modeling, is a complex task even if conducted by single learners. Current research is far from having solid and detailed results about the cognitive and social processes involved in collaborative modeling activities, which could in turn inform decisions about "success factors" for using flexibly shareable dynamic models in educational contexts. We cannot expect an automatism that leads to successful collaborative inquiry learning just based on highly flexible tools being available to educational practitioners. It may be entirely possible that the sharing of representations does not always lead to fruitful collaborative learning scenarios — e.g. Margaritis *et al.* (2003) doubt this and only allow for full model sharing in the ModellingSpace application. However, the positive experiences of the small case study reported in this article suggest to re-think the answer to this question.

Certainly, more empirical research is needed to inform concrete guidelines and recommendations for designing collaborative modeling tools and learning scenarios that finally "work" and use the high degree of options available in modern systems in a pedagogically reasonable way. Exploring the potential benefits of the various modeling support types that technology can offer and that were discussed in this paper (including different forms of sharing models and different activity structures) is on our research agenda. The availability of highly flexible, interoperable and adaptable tools, as presented in this article, is a prerequisite for conducting these empirical studies.

Therefore, we believe that substantial parts of an inquiry learning process can be enabled by the Cool Modes system, when the different collaboration modes are used in combination with our macro level approach. An important aspect of this approach is that it also works beyond Cool Modes — in principle, the "remote control" we propose is capable of steering, and thus integrating, multiple VLEs with their specific functions.

References

Ainsworth, S. (1999). The functions of multiple representations. *Computers and Education*, *33*(2–3), 131–152.

Armstrong, W. (1972). *The ability of fifth and sixth graders to learn selected topics in probability*. Unpublished Ed.D. Thesis, University of Okalahoma.

Bredeweg, B., & Forbus, K. (2003). Qualitative modeling in education. *AI Magazine*, *24*(4), 35–46.

De Jong, T., & van Joolingen, W. (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research*, *68*(2), 179–201.

Fischbein, E. (1975). *The intuitive sources of probabilistic thinking in children*. Dordrecht, The Netherlands: D. Reidel Publishing Company.

Forrester, J. (1968). *Principles of systems*. Waltham, MA: Pegasus Communications.

Harrer, A., Malzahn, N., Hoeksema, K., & Hoppe, U. (2005). Learning design engines as remote control to learning support environments. *Journal of Interactive Media in Education*, Special Issue on Advances in Learning Design (August 2005), http://jime.open.ac.uk/2005/05/.

Heiler, S. (1995). Semantic interoperability. *ACM Computing Surveys*, *27*(2), 271–273.

Hernandez, D., Asensio, J., & Dimitriadis, Y. (2004). IMS learning design support for the formalization of collaborative learning patterns. In Kinshuk *et al.* (Eds.), *Proc. 4th Int. Conf. on Advanced Learning Technologies* (pp. 350–354). Los Alamitos, CA: IEEE Press.

Hoppe, U. (2004). Collaborative mind tools. In M. Tokoro, & L. Steels (Eds.), *A learning zone of one's own — sharing representations and flow in collaborative learning environments* (pp. 223–234). Amsterdam, The Netherlands: IOS Press.

IMS LD (2007). Learning Design Specifications http://www.imsglobal.org/learningdesign/index.cfm.

Jonassen, D. (2000). *Computers as mindtools for schools*. Upper Saddle River, NJ: Prentice Hall.

Jonassen, D., Peck, K., & Wilson, B. (1999). *Learning with technology: A constructivist perspective*. Columbus, OH: Prentice Hall.

Kay, A., & Goldberg, A. (1977/2001). Personal dynamic media. In R. Packer & K. Jordan (Eds.), *Multimedia — from wagner to virtual reality* (pp. 167–178). London, England: W.W. Norton & Company.

Konold, C. (1991). Understanding Students' beliefs about probability. In E. Glaserfeld (Ed.), *Radical constructivism in mathematics education* (pp. 139–156). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Konold, C. (1995). Issues in assessing conceptual understanding in probability and statistics. *Journal of Statistics Education, 3*(1), http://www.amstat.org/publications/jse/v3n1/konold.html.

Kurtz dos Santos, A., & Ogborn, J. (1994). Sixth form student's ability to engage in computational modeling. *Journal of Computer Assisted Learning*, *10*(3), 182–200.

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

Margaritis, M., Fidas, C., Avouris, N., & Komis, V. (2003). A peer-to-peer architecture for synchronous collaboration over low-bandwidth networks. In K. Margaritis & I. Pitas (Eds.), *Proceedings of the 9th Panhellenic Conference in Informatics* (pp. 231–242). Thessaloniki, Greece.

Miao, Y., Hoeksema, K., Hoppe, U., & Harrer, A. (2005). CSCL scripts: Modelling features and potential use. In T. Koschmann, D. Suthers & T. W. Chan (Eds.), *Proceedings of the International Conference on Computer Supported Collaborative Learning* (pp. 426–432). Mahwah, NJ: Lawrence Erlbaum.

Milrad, M., Spector, M., & Davidsen, P. (2002). Model facilitated learning. In S. Naidu (Ed.), *Learning and teaching with technology: Principles and practices* (pp. 13–27). London, England: Kogan PagePublishers.

Or-Bach, R. (2003). Design consideration for supporting collaborative modeling. In V. Devedzic, J. M. Spector, D. G. Sampson, & Kinshuk (Eds.), *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies* (pp. 219–223). Los Alamitos, CA: IEEE Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Perkins, D. (1991). Technology meets constructivism: Do they make a marriage? *Educational Technology*, *31*(5), 18–23.

Petri, C. (1962). *Kommunikation mit Automaten*. Bonn, Germany: Schriften des Rheinisch-Westfälischen Instituts für Instrumentelle Mathematik.

Pinkwart, N. (2005). *Collaborative Modelling in Graph Based Environments*. Berlin, Germany: Dissertation.de — Verlag im Internet.

Read, T., Verdejo, F., & Barros, B. (2003). Incorporating interoperability into a distributed elearning system. In D. Lassner & C. McNaught (Eds.), *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications* (pp. 273–282). Norfolk, VA: AACE.

Roschelle, J., DiGiano, C., Repenning, A., Phillips, J., Jackiw, N., & Suthers, D. (1999). Developing educational software components. *Computer*, *32*(9), 50–58.

Schwarz, B. B., & Glassner, A. (2007). The role of floor control and of ontology in argumentative activities with discussion-based tools. *International Journal of Computer-Supported Collaborative Learning*, *2*(4), 449–478.

Sierhuis, M., & Selvin, A. (1996). Towards a framework for collaborative modeling and simulation. In A. Selvin & M. Sierhuis (Eds.), *Proceedings of the workshop on "Strategies for collaborative modeling and simulation" at the the 5th ACM Conference on Computer Supported Cooperative Work*.

Suthers, D., Weiner, A., Connelly, J., & Paolucci, M. (1995). Belvedere: Engaging students in critical discussion of science and public policy issues. In J. Greer (Ed.), *Proceedings of the 7th World Conference on Artificial Intelligence in Education* (pp. 266–273). Charlottesville, VA: Association for the Advancement of Computing in Education.

Tsovaltzi, D., Rummel, N., McLaren, B. M., Pinkwart, N., Scheuer, O., Harrer, A., & Braun, I. (2010). Extending a Virtual Chemistry Laboratory with a Collaboration Script to Promote Conceptual Learning. *International Journal on Technology Enhanced Learning*, *2*(1/2), 91–110.

van Joolingen, W. (2000). Designing for collaborative discovery learning. In G. Gauthier, C. Frasson & K. VanLehn (Eds.), *Proceedings of the 5th International Conference on Intelligent Tutoring Systems* (pp. 202–211). Berlin, Germany: Springer.

van Joolingen, W., de Jong, T., Lazonder, A., Savelsberg, 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.

van Joolingen, W., & Löhner, S. (2001). Representations in collaborative modeling tasks. In S. Ainsworth *et al.* (Eds.), *Proceedings of the workshop on "External representations in AIED: Multiple forms and multiple roles" at the 10th International Conference on Artificial Intelligence in Education*. http://www.psychology.nottingham.ac.uk/research/credit/AIED-ER/vanjoolingen.pdf.

Vogten, H., Koper, R., Martens, H., & Tattersall, C. (2005). An architecture for learning design engines. In R. Koper & C. Tattersall (Eds.), *Learning Design* (pp. 75–90). Berlin, Germany: Springer.

White, B., & Frederiksen, J. (2000). Technology tools and instructional approaches for making scientific inquiry accessible to all. In M. Jacobson & R. Kozma (Eds.), *Innovations in Science and Mathematics Education* (pp. 321–359). Mahwah, NJ: Lawrence Erlbaum.

Wild, M. (1996). Mental models and computer modeling. *Journal of Computer Assisted Learning*, *12*(1), 10–21.