CML – The ClassSync Modeling Language

John Brecht, Mark Chung, and Roy Pea

The Center for Technology in Learning, SRI International

john.brecht@sri.com mark.chung@sri.com roy.pea@sri.com

ABSTRACT

The ClassSync Modeling Language (CML) addresses the problem of creating a controlling overlay to classroom learning activities, or e-learning workflows. Our aim is to allow authors and teachers to generate a mapping from activity design to its implementation in a wirelessly networked classroom with ubiquitous use of handheld computers for information exchange. CML models e-learning workflows with three major components: actors, data objects, and interaction networks. Actors are the diverse performers of actions, data objects are the semantically typed information units that are made available for exchange. Interaction networks are virtual networks constructed on top of whatever network ClassSync is running on, and dictate how information may flow through the ClassSync system (from actor to actor). Activities are the processes performed via these three components, in which actors create and consume data objects and exchange them over an interaction network. The benefits of this approach for students, curriculum designers, teachers, publishers and learning sciences researchers are highlighted.

Keywords

EML (Education Modeling Language), handheld devices, wireless networking, instructional discourse, classroom workflow

NEED

The design and performance of instructional activities is a fundamental problem for research in interactive learning environments. These issues are particularly problematic for the teacher wishing to implement various CSCL scenarios such as small group learning, although more common patterns of classroom interaction, such as the familiar whole-class IRE pattern (Initiation-Response-Evaluation, as in Mehan, 1978), can also be demanding. In non-computer based collaborative patterns of instruction, teachers assemble plans for assigning students to groups, distribute written or oral instructions for their work, including assignment of specific roles to students and constraints on their work in terms of time, document access, and work products required to result from their activity. Teachers collect work, annotate it with scores or comments, and re-distribute it to students to further their learning. Dozens of learning activity cycles like these a day define the work of the classroom.

Designing, implementing, and monitoring these learning activities is a workflow challenge, as research in teacher cognition, planning, and learning has highlighted (e.g., Borko & Putnam, 1996; Cohen et al., 1993; Hammond & Sykes, 2001; Little & McLaughlin, 1993). Teachers currently plan and manage these efforts using a broad array of documents and physical resources, including class lists, publishers' teacher activity guides, seat-based distribution of textbook instructions or photocopied materials, physical separation of groups in the classroom space, worksheets to be completed, informal observations of equitable participation by students in discussions, and the like. In the case of small-group learning, tracking group progress and individual students' participation is important, given well-known problems in group work such as the "free rider" phenomenon (Salomon & Globerson, 1989), and the likelihood that student work in groups can be beset with collaboration breakdowns (Barron, 2000) and unproductive inquiries if interim project milestones are not required (Polman, 1999).

These classroom activity structures and affiliated workflow patterns have become yet more complex with the introduction of computer technologies in classrooms, in which individual use of shared resources such as desktop computers, printers, computer projectors for displaying work, data collection probes and sensors, and other devices need to be integrated with non-computer facilitated classroom activity components.

Furthermore, much of the effort required to set up such arrangements of learners, documents, and task structures, much less to learn from the outcomes of such designs in ways that could lead to improvements, does not result in searchable records and re-usable activity structures. We call this the 'act becomes artifact' problem—and opportunity for innovation and research. With increasing teacher accountability for promoting student learning, it will become increasingly important to provide support for ongoing formative assessment to help diagnose student difficulties and determine productive strategies for overcoming them. Semantic tagging of the actors, data objects, and interaction networks has significant promise for "informating" (rather than "automating": Zuboff, 1988) classroom workflow by turning normally transient instructional (and learner) acts into artifacts for data mining.

Traditional desktop-based Instructional Learning Systems (ILSs), that pre-define activity and response frameworks for individuals are not up to meeting the challenges of learning workflow needs for diverse activity structures among teachers and students (Roschelle et al., 2001). Schwartz et al. (1999) and Barron et al. (1998) highlight the importance for learning environments of flexibly adaptive instruction and formative assessment, in which the teacher deals contingently with the emerging needs of learners and groups to provide constructive learning opportunities.

We envision an extremely low cost system with a wireless network, individual handheld computers, minimal maintenance during school, and with low-threshold user interface for teacher and student use to handle the major proportion of instructional workflow. Real value needs to be added to the paper now primarily used for these purposes. Such a system needs to handle access permissions, distribution of tasks and instructional resources, and collection of activity results from students without requiring teachers to become network system administrators. HotSyncing (in the manner of Palm OS synchronization of desktop and palmtop computer contents) does not logistically meet our needs because the model of coordinating desktop and handheld devices does not support the just-in-time quality that classroom workflow requires.

APPROACH

CML, the ClassSync Modeling Language, is a language for generating a mapping from activity design to implementation of the activity on a network of handheld computers. CML has three components: actors, data objects, and interaction networks. *Actors* are the performers of action in the system, including people (students or teachers, coupled with their devices), groups of actors (where each actor has a well-defined group role), and computer agents called "bots" which help manage the system. *Data objects* are the information units in the ClassSync system and include artifacts such as media, messages, records, and processes (which may control tools for creating or modifying such artifacts or refer to them). Each data object may be classified in terms of one or more semantic types. *Interaction networks* are virtual networks constructed on top of whatever network ClassSync is running on. Interaction networks dictate ways in which information (data objects) may flow through the ClassSync system (from actor to actor). Activities are the processes performed via these three components. An activity will be a process in which actors consume and create data objects and exchange them over some interaction network.

CML is a modeling language that will allow authors and teachers to construct activities by creating assemblages of these component elements. Once such an activity has been modeled in this way, it is up to the ClassSync system to implement an activity at runtime. It is our intent that the CML description will allow for a wide range of implementations, on a wide range of technical platforms. Our goal is that CML will become useable in real time for assembling new activities "on the fly" as new learning and teaching opportunities emerge, as well as a means of preparing activities ahead of time by the teacher or other educational agents.

A CML SCENARIO

To introduce CML, we first present a usage scenario and later explore CML components in more detail.

A User Interface for Flexible Activities "At the Board"

Imagine a user interface for the teacher supporting activities "at the board," in which one or more students are called upon to solve a problem in view of the class. This interface uses a theatrical metaphor. The theater has a stage, an audience, a script, and a backstage area where the props are stored. The script is a list of scenes specifying the cast (group definition of actors), situation (activity), and props (data objects). The teacher acts as director and calls actors, represented by icons, onstage from the audience. Once onstage, the actors have control of the props. The audience has a view of the action on stage. (The stage and actors model will be familiar from Programming by Rehearsal (Finzer & Gould, 1984), Stagecast (Smith & Cypher, 1998), and other Xerox PARC-influenced approaches to developing a computer program. We find the metaphor apt but not our point.)

Running the Show – IRE (Initiation-Response-Evaluation)

Mehan (1978) defined a now well-recognized classroom discourse structure in the Initiation-Response-Evaluation sequence, which characterizes many instructional scenarios as: a teacher initiates an instructional sequence by asking a question, a student responds, and the teacher evaluates that response. So, let's follow the production of a CSCL variant of an IRE scene in our theater. Suppose our classroom is about to participate in GLOBE, a worldwide program for primary and secondary school students to collect, analyze, and report earth science data (GLOBE, 2001; Means & Coleman, 2000). The students have been instructed that they will be working together in groups to collect and analyze data. In order to ensure that the students understand this process well, the teacher will call upon some students to perform a measurement and analysis activity in front of the classroom. Collectively, they will measure the air temperature, pressure, and humidity and compare those values with similar measurements from another classroom across the country, and a set made at the same school in the previous semester.

First, the teacher initiates the scene by clicking "casting call" on it in the script. The Casting bot asks everyone who is to be invited to "try out" (in this case the entire class) by sending a message to the class group manager, which relays the message to every member of the class. The content of this message is a solicitation with the scene description ("You will work with a group of students to collect and analyze data...") and a list of the roles that may be volunteered for ("Analyst, Temperature Measurer, Pressure Measurer, etc..."). The Casting bot passes responses back to the teacher. The teacher's GUI highlights the icons for the students who volunteered ("raised their hands"). (Some equity-related statistics may pop up as well, such as the time since the student last raised their hand or the total number of times the student has raised their hand on that day and overall.)

Next, the teacher picks students to participate in the scene by dragging their actor icons onstage to the positions corresponding to the roles they should play. When the selection is complete the teacher clicks "positions." The Casting bot then adds the selected students to the "cast" group for the scene. A Stage Director bot moves props from backstage (the teacher's repository) to front stage (the class group repository), according to the script. The cast group manager notifies students as they are added to the cast group, tells them where the props are and how to control them and tells them what their role is in the group. In this case, the "prop" the analyst receives is a spreadsheet containing the data to which the new data are to be compared. The other students in the activity get physical props—probes to attach to their handhelds.

The teacher now clicks "lights," a command giving the class group view permission on the props and on the cast group manager itself, and allowing each member of the class group to see the cast group manager's member list. The GUI on the students' devices then provides each student with a view of the props, and a list of the cast—the illuminated stage. Having a "view" of a prop means that the prop can be viewed but not controlled by the viewer. For instance, the entire class could look at the data in the analyst's spreadsheet, but could not change it or add to it.

Finally, the teacher clicks "Action", which gives the cast group control of the props (and thus permission to modify them). Then the cast may begin to perform the scene by controlling the props and communicating with each other. As they do so, the audience's views are updated appropriately. So, in our GLOBE scenario, the action begins with the three measurers making their measurements. As each one performs the specified steps for these subactivities within the overall activity, the other students will have a view of what is going on in the measurers' devices. One by one, they transmit their measurements to the analyst who plots the data. When they complete these tasks, the group members prepare a summary by editing a shared summary document viewable by the class.

The teacher may end the scene by dismissing the actors (casting bot removes them from the cast group) and dropping the lights (view permission taken away from the class group.) Before the lights are dropped, of course, it is likely that the teacher will want to take the stage and discuss the final state of the props of the class, or even modify them further. Likewise, the teacher may, at any time, add him or herself to the group onstage, and manipulate the props in some manner with them. For example, the GLOBE teacher may review and annotate the summary document before the class, or jump in during the analysis phase to demonstrate the procedure for generating a scatter plot in the spreadsheet application.

ACTIVITIES IN CML

Definition

At a modeling level of description, classroom activity can be defined as an information processing function having an input, an output, and a process by which the output is to be generated from the input This functional definition of an activity can be used to model activities ranging from IRE sequences, to role-playing activities (such as participatory simulations), to small group project-based work, to seatwork or homework exercises.

Linked collections of activities

To achieve this versatility we allow for activities to be chained, networked, and nested. We call such a group of activities a *linked* activity. In the case of a chained activity, the output of one activity process becomes the input of another. For instance, one step in a sequence of a scientific activity might be to do data collection and a next step of the activity might be to do analysis of the data. So the output of the first step, the parameters and values of the collected data, becomes the input of the second step, whose output, in turn, could be a report of the analysis.

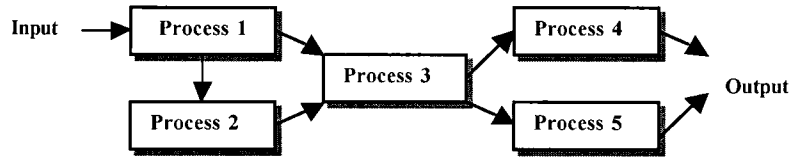

Figure 1: A networked activity

A networked activity (Figure 1) is a more general case of a chained activity, but now there are branches in the chain. At a branching point, the next process is chosen based on the outcome of the current activity. Note that Process 4 and Process 5 both produce the same output. How is it that two different processes can lead to the same output? When activities are defined in CML, they are defined as having one or more specifiable semantic types of input and one or more specifiable semantic types of output. The specific value for the input does not necessarily have to be chosen until the activity is assigned. For instance, imagine the input of the activity diagrammed in Figure 1 is defined as *triangle* and the output is defined as *area*. This means that when the problem is assigned to a student, a specific triangle must be supplied, and a value representing an area will be the output that is ultimately produced by the activity (regardless of the path through the network). Inputs and outputs are data objects. We will discuss data objects and their associated semantic types in greater depth below.

Nesting activities provide a means by which a chained or networked activity is encapsulated in one activity, which in turn forms a part of another linked activity. So long as the linked activity has one input and one output it can be treated in CML just as an ordinary "one step" activity for the purposes of networking. By allowing activities to be nested in this way we make it easy to reuse activities and allow for designers to work on many levels of granularity.

Activity Lifecycles

The simplest lifecycle for an activity to have is to be transitory. The activity is assigned, the student immediately begins work on the process, and the activity ends when they generate the output. However, networks of activities can take on a more dynamic character. Consider the networked activity portrayed in Figure 1. Now, rather than one student making their way through the network, imagine instead that each activity in the network has been assigned to a different student. Further, rather than each activity in the network being performed once, instead imagine that an assigned activity *lingers*. In this situation, each activity is performed whenever the student to which it has been assigned receives the input associated with the activity.

In addition to activities lingering, ClassSync allows for *simultaneity* in the system. By this, we mean that there may be multiple data objects moving through the system simultaneously. In fact, ClassSync allows more than one input to be "processed" by an activity simultaneously. Imagine the class is engaged in a role-playing exercise in which they play the parts of legislators passing laws. It might be interesting in this exercise for a legislator to have more than one bill on their desk at a time. A simpler example might be a common situation where individual students do basic research on some set of topics, and then as a group, they bring their reports together to generate an overall report based on all the individual reports.

The third dynamic quality of ClassSync is *contingency*. Contingency has already shown up implicitly in our networked activity with branching. By contingency we simply mean that activities are not necessarily performed. Rather, they are only performed if the appropriate input data object is sent to the actor to whom the activity has been assigned.

Authors specify when activities end, and can do so in a number of ways (with the teacher having the option to override the end condition). One way is similar to our transitory activity. As soon as the activity produces its final output, the activity ends. Of course, the author may intend that the students perform the activity process a few times, generating multiple outputs. In this case, there are a number of options. There can be an explicit time limit, at the end of which any existing intermediary or final output is collected. Another end condition is that the activity ends after all the required inputs have been supplied, and an output has been generated for each input.

COMPONENTS OF CML

Before we can continue to discuss how activities are created and used in practice, we must digress to introduce the components of CML—actors, data objects, and interaction networks.

Actors

While activities are the processes performed in the ClassSync system, actors are the entities in the system that either perform these processes or that facilitate their performance. CML includes three kinds of actors: *person, group manager,* and *bot*. A person is one individual and his or her associated device. A teacher is a special case of a Person. The teacher is the "superuser" of the ClassSync system. A group manager is the hub of an interacting group of actors (persons or other groups), primarily characterized by the interaction topology it enables amongst the members of the group. A bot is a computer agent, capable of performing specific tasks and of communicating its capabilities. Actors have three categories of properties associated with them: descriptors (metadata), data, and one or more transient states. These properties are summarized in Table 1 below. (All actors have the properties listed in the first row.)

Table 1: Properties of CML Actors

Actor	Descriptors	Data	Transient states
All	• ID, address repository	• List of associated data objects	• Transfer Queue[1] • Linkage[2]
Person	• e.g., gender • e.g., academic level	• List of group affiliations as ordered pairs: (group ID, role(s) played in group) • List of activities	• Activity progress
Group Manager	• Interaction topology • Purpose/type	*As per Person, plus* List of members as ordered pairs: (actor, role)	• Activity progress
Bot	• Description of capabilities[3]	• Schedule	• Current task

Data Objects

Data in ClassSync system are encapsulated as data objects, as described by the Model-View-Controller architecture pioneered by Smalltalk-80™. A data object has a set of semantic types and parent/child relationships associated with it. The semantic types serve to identify in what ways the data object can be used in activities. Parent/child relationships are used to create collections of related objects. Data objects may be shared (simultaneously viewed or edited) by multiple actors in a group, as well as transferred from actor to actor.

Interaction Topologies

Actors may be members of various groups, but they may not interact (perform data object transfers, including messaging) with them until they are linked to the members of the group and/or its manager. The first step in beginning group interaction is to open a link to the group manager. What happens next depends on the topology of the group's network. Regardless of network topology, it should be possible for an actor to send a data object to every member of the group in one step (mass mailing.) Likewise, once linked to the group manager, it should also be possible to request data from it, such as its member roster and present linkage.

[1] A list of data objects currently scheduled for transfer between this actor and one or more others.

[2] A list of all links to or from the actor, where to be "linked" effectively means to be "logged into," or "able to perform transfers with.

[3] Implemented using the Open Agent Architecture (Martin, David L. and Cheyer, Adam J. and Moran, Douglas B. 1999).

Client/Server Networks

If a group interoperates via a client/server network, the group manager acts as "server" for the group. Recall that, like all actors, there is a repository associated with the manager. This repository will serve as the shared "file space" for the group. Here, group members may post data objects that they wish others to download. Also, any data objects that are shared by the group will reside here.

Peer to Peer Networks

If the network topology is P2P (Peer to Peer), then logging in to the group manager should result in a link being created to each other member of the group. There is no "server" in this case. If a file is to be shared, a group member must host it on his or her own device. In fact, there is no single group manager in the case of P2P. A copy of the group manager will reside on every device. This distributed group manager must synchronize itself. For example, when an actor logs in to the group, the group manager on that actor's device must go out onto the P2P network and find the other managers, and set its state to their state.

A special case of a P2P network is a limited and/or directional P2P network, in which an actor links to some subset of the group (limited), and a given actor may only transfer in one direction with some other actor in the group (directional). This could be used to form a chain of actors in which data objects travel in one direction.

CML IN ACTION: BRINGING A LEARNING GROUP TO LIFE

Groups

Creating a group

As mentioned earlier, the teacher is the superuser of the ClassSync system, which gives the teacher the ability to create groups. The creation of groups has two stages—defining the group, and enabling the group. Once the group has been defined and enabled, students can participate in the group. To define a group, the teacher creates a group manager for the group. This act has two steps, assigning descriptors for the manager, and assigning it initial data.

There are four types of data associated with the group manager—data objects, a member roster, group affiliations, and a list of activities. The teacher creates a list of Universal Resource Names (URNs) of the data objects that the group should have at the start. Then the teacher creates a list of roles to be played by members of the group. Finally, if the group is to be a member of a larger group, the teacher identifies that group and assigns the subgroup's role in the supergroup.

Enabling a group

The teacher enables the group after it has been defined, and the system performs setup according to the group topology. For a Client/Server group, the system creates a group manager repository containing copies of the data objects assigned to the group, and schedules a bot to locate the students to add to the group. The division of students into groups can be done randomly, strategically (see Example), or according to a specific list. For a P2P group, there is no centralized repository, so the bot copies the group manager to each member and copies the data objects assigned to the group to one of the group member's repositories.

Participating in a Group

The system notifies students when they have been placed in a group. Students link to the manager to join the group and learn what their role is (this can require a login with a password). They can also find out who the other members are and whether they're logged in, what the group's data objects are, and what the group's goal or role is. Furthermore, they may also find themselves linked to the other members of the group if the group is a P2P group.

"Jigsaw" Groups

In the spirit of the "Jigsaw Classroom" approach (Aronson & Patnoe, 1997), a CML implementation could automatically create groups that join everyone whose roles in their existing groups are the same, with some default interaction topology. The teacher managing the system controls whether or not this function is turned on, and what the topology of an automatically generated jigsaw group ought to be.

Example

After using the IRE to demonstrate the work the students will be doing, our teacher begins the GLOBE activity by dividing the 30-person class into six groups and assigning two groups to each of the three measurement protocols. Each group of five has the following roles, filled by actors of type "Student": data collectors (3), data analyst (1), and a reporter (1) who compiles the group's write-up. The teacher specifies a group topology of P2P, though that may change when activities are assigned. Additionally, there is a group of groups called "Overall GLOBE Investigation" that includes all six groups. This group has six roles, two for each protocol, and each role must be filled by an actor of type "Group" with the appropriate description, e.g., "Humidity Investigators."

Next, the teacher must assign specific actors to roles. The teacher can do this by gestural input, or the system may offer some automated process operating under a teacher or author defined strategy. When assigning students to roles the teacher, or the automated process, might make use of actor descriptors. For example, it may create groups at random, but with the caveat that each group have gender balance. In the GLOBE activity example, the teacher may elect to assign the "natural leaders" in the class to the role of reporter. Also, in anticipation of the activity to come, the teacher in this activity elects to automatically generate a jigsaw group with P2P topology amongst all the data analysts.

Activities

Creating Activities

Activities are defined as having an input, an output, and some process by which the output should be generated from the input (e.g., using a tool controlled by CML or referred to by it, but outside the system). To create an activity with CML, input, output and process must be specified. Input and output are defined by specifying a semantic type of the data object. To specify process, the author must include a data object that provides instructions

and, optionally, specifies the role of the actor who should do the activity. The process definition must also include sufficient information for the system to determine where to send the output data object when it is completed.

We differentiate between an activity definition, and an actual activity instance. Thus, a teacher can define an activity as having a certain type of input data object, a certain role of actor, and, a certain type of output data object. However, it isn't until assignment (or runtime, in programmer-speak) that specific values are filled in and the activity is instantiated.

Creating Linked Activities

A linked activity still has the same properties as a normal activity—input, process, and output. As mentioned earlier, the input is simply the input of the first subactivity, and the output is the output of the final subactivity. The actor associated with the process, however, must be a group if the subactivities are to be performed by multiple actors. (Otherwise, the actor is just the one actor that performs all of the activities.) Besides the group associated with the superactivity, there is the collection of activities itself and its link topology. The author must identify what these activities are and which links to which. Bear in mind, that we are still only defining the activity, not instantiating it, so this mapping of actors to activities is still in the abstract. That is, we are still only identifying *roles*, not specific actors.

Assigning Activities

ClassSync automates assigning activities to actors so that the process is the same regardless of the type or complexity of the activity. The teacher simply picks the activity, an input data object, and a group to which the activity is to be assigned. The assignment process then goes and finds each actor that is a member of the specified group that has the capability to do the activity. Having a "capability" simply means that the actor is of the appropriate role (or roles) within the group.

Example Activity Assignment

In our GLOBE classroom, the investigation groups have been created, and now it is time to assign the investigation activities. The teacher has created data collection activities, an analysis activity, a write-up activity, and a discussion activity. The discussion activity is to be assigned to the "Overall GLOBE investigation" and its input is a data object that is composed of a union of all the collected data (which will be produced by the analysts). The data collection, analysis, and write-up activities compose a network activity called "X Investigation," where "X" is one of the three protocols (air temperature, pressure, humidity).

To assign the activities, the ClassSync system must, for each activity, identify every actor in the system that has the role associated with the activity. For instance, there is an activity for measuring temperature. The role associated with the activity would be "'data collector' who is a member of a group whose type is 'Temperature Investigators.'" (Note that we can define roles in ClassSync iteratively—"an X, which is a member of a Y, which is a member of a Z...") Altogether, the system should find six such actors, but it doesn't matter that we know this is the case ahead of time. Without changing anything about the activities, the teacher should be able to alter the number and type of actors in the class, and the system should still be able to handle it.

Once an actor is identified for assignment, the ClassSync system adds the activity to their activity list and, if appropriate, schedules a transfer with them to give them an input data object. When the assignment is made, the student or group should receive notification about the assignment. Once all the assignments have been made, the teacher indicates that the students may begin work using some start trigger associated with the overall activity. This trigger may be an initial input data object for the entire network of activities, or it may simply be a message to the top level group in the activity network telling it to "begin."

CLASSSYNC SUPPORTS COLLABORATIVE PROJECT BASED LEARNING

We have described in this paper how CML provides the infrastructure essential for collaborative project-based learning by coordinating the activity lifecycle, providing group administration services, and managing resources and communication processes. CML provides a language that specifies dynamic configurations of this infrastructure. Teachers interact with ClassSync at a high level by creating groups and assigning activities to them. ClassSync automates the details of creating a group by notifying students that they now belong to a group, by making resources available to the group, and enabling resource sharing and messaging. ClassSync automates the details of activity assignment, assigns roles to members, and transitions to the next activity when the activity completes.

CML is consistent with Activity Centered Design

Activity Centered Design (Gifford & Enyedy, 1999) represents a shift in the theoretical framework of CSCL from Learner Centered Design—which proves less suitable for collaborative models because of its focus on the

individual—to a model in which learning happens within an activity system consisting of people, artifacts (tools and data objects, as per above), and tasks linked within a social context. The model is neither simply learner-centered nor teacher-centered; rather, learners draw upon resources such as the teacher, other students, or tools and data as they participate in an activity. We applaud this emphasis as a productive modeling framework for CSCL as well as the other forms of socially-situated and artifact-mediated instructional activity that take place in classroom workflows (e.g., Cole, 1996). Such an Activity Theoretic focus has its roots in work by Vygotsky (1978) and Leont'ev (1979), and has been used fruitfully in CSCW research (e.g., Bodker, 1997; Nardi, 1996).

CML fits neatly within this framework by providing a language for expressing the relations between the activity, actors, and tools/data, which operate as part of a dynamic system. The system is dynamic because the relations do not have to be fixed or tightly coupled, nor do the activities have to proceed on a fixed trajectory. Actors move from one activity to the next, have tools and data objects at their disposal, and create data objects/tools that they may share or exchange with other actors.

WHO BENEFITS FROM CML?

It is also worth highlighting what we consider to be the primary benefits from use of the ClassSync Modeling Language, for curriculum designers, students, teachers, publishers, and researchers in the learning sciences.

- Curriculum designers benefit from CML by having a means of expressing activities covering a wide range of interaction scenarios. CML provides a level of abstraction above the hardware and network, so a designer can specify activities that could run in a variety of settings.
- Students benefit because ClassSync gives them the ability to participate in a number of interaction topologies. ClassSync simplifies the technical hurdles of transitioning between these topologies through automatic configuration and resource management for easy access to information and people resources.
- Teachers benefit because ClassSync coordinates classroom workflow and simplifies the task of grouping students and managing the details of assigning work to groups. ClassSync enables real-time performance measurement and post-class playback of activity sequences for classroom reflection or teacher professional development purposes.
- Publishers can use CML tools to address the problem of scaling to cover entire curricula with authoring tools and reusable activity structures. CML, as a runnable formalism, also is testable, so that activities can be run in a simulated environment for quality assurance.
- Researchers in the learning sciences benefit from CML because activities may be instrumented to monitor and record the learning process; later, data mining techniques may be applied to the collected data, and results fed back to support the teacher's instructional decision-making.

CHALLENGES

We have outlined a specification for the ClassSync Modeling Language, but acknowledge that there are formidable challenges to implementing a working system and applying it to commonplace curriculum design and classroom use.

- Every modeling language faces a tradeoff between expressiveness and usability. As language complexity increases, authors may create richer activities, but the system may become unwieldy. We are seeking a workable balance in this tradeoff space that nonetheless will provide a powerful action augmentation framework for teachers.
- CML does not model everything of significance to learning interactions that occurs in a classroom (e.g., social exchanges, uses of tools that are not computer-controllable). There will always be a gap between system knowledge and tacit knowledge, between formal interactions with the devices and real-world interactions. Nonetheless, we expect CML can model centrally significant aspects of e-learning workflows.
- The ClassSync system may require a non-trivial amount of training for the teacher and students, who may not be familiar with handheld devices, much less information exchange over a wireless network. We expect a design research focus (e.g., Edelson, Gordin & Pea, 1999) can iteratively improve on such issues toward a readily learnable system.
- Any implementation of ClassSync is bound to face practical issues related to the particular hardware or network used. This may lead to significant differences in system capability or performance that impose constraints on the kinds of activities that are practical.

We anticipate that in our ongoing work to successfully develop CML, we will be applying iterative design techniques, involving authors and teachers in the design of the language and the authoring tool. We will need to conduct field tests of the resulting activities to verify that the language is sufficiently expressive, and develop a test framework for authors to use during development. We are employing a bootstrapping process (Engelbart, 1962) in

which we use the products of our development and conduct synergistic activities that focus on specific interaction topologies and activities in specific domains as steps towards a complete system.

ACKNOWLEDGMENTS

Thanks to the many contributors from the WILD (Wireless Internet Learning Devices) project team at SRI International: Tristan de Frondeville, Chris DiGiano, Sarah Lewis, Judy Li, Charlie Patton, Jeremy Roschelle, Deborah Tatar, Phil Vahey, and Wenming Ye. We gratefully acknowledge support from SRI International and the National Science Foundation.

REFERENCES

Aronson, E., & Patnoe, S. (1997). *The jigsaw classroom: Building cooperation in the classroom (2nd ed.)*. New York: Addison Wesley Longman.

Barron, B. (2000). Achieving coordination in collaborative problem solving groups. *Journal of the Learning Sciences*, *9*(4), 403-436.

Barron, B., Schwartz, D.L., Vye, N.J., Moore, A., Petrosino, T., Zech, L. & Bransford, J.D. (1998). Doing with understanding: Lessons from research on problem and project based-learning. *Journal of the Learning Sciences*, *7*, 271-311.

Bodker, S. (1997). Computers in mediated human activity. *Mind, Culture and Activity*, *4*(3), 149-158.

Borko, H., & Putnam, R. T. (1996). Learning to teach. In D. C. Berliner & R. C. Calfee (Eds.), *The handbook of educational psychology (pp. 673-708)*. New York: Macmillan Publishing.

Cohen, D. K., McLaughlin, M. W. & Talbert, J. E. (1993). (Eds.), *Teaching for understanding: Challenges for policy and practice*. San Francisco, CA: Jossey-Bass.

Cole, M. (1996). *Cultural psychology*. Cambridge, MA: Harvard University Press.

Edelson, D. C., Gordin, D.N., & Pea, R. D. (1999). Addressing the challenges of inquiry-based learning through technology and curriculum design. *Journal of the Learning Sciences*, *8*(3&4), 391-450.

Engelbart, D.C. (1962, October). *Augmenting human intellect: A conceptual framework* (Summary Report, AFOSR-3233). Menlo Park, CA: SRI International.

Finzer, W. & Gould, L. (1984). Programming by rehearsal, *Byte*, *9*, 187-210.

Gifford, B.R. & Enyedy, N.D. (1999). Activity-centered design: Towards a theoretical framework for CSCL. In C.M. Hoadley & J. Roschelle (Eds.), *Proceedings of the Computer Support for Collaborative Learning (CSCL) 1999 Conference (pp. 189-196)*. Palo Alto, CA: Stanford University [Available from Lawrence Erlbaum Associates, Mahwah, NJ]

GLOBE Program (2001, January). *The GLOBE Program.* Washington, D.C. (www.globe.gov)

Hammond, L. D. & Sykes, G. (2001). (Eds.), *Teaching as the learning profession: Handbook of policy and practice.* San Francisco: Jossey-Bass.

Leont'ev, A.N. (1979). The problem of activity in psychology. In J. Wertsch (Ed.), *The concept of activity in Soviet Psychology*. New York: M.E. Sharpe, Inc.

Little, J.W. & McLaughlin, M. (1993). (Eds.), *Teachers work.* New York: Teachers College Press.

Martin, David L. and Cheyer, Adam J. and Moran, Douglas B. (1999) The Open Agent Architecture: A Framework for Building Distributed Software Systems. *Applied Artificial Intelligence, vol. 13, no. 1-2, pp. 91-128, January-March 1999.*

Means, B., & Coleman, E. (2000). Technology supports for student participation in science investigations. In M.J. Jacobson & R.B. Kozma (Eds.), *Innovations in science and mathematics education (pp. 287-319).* Mahwah, NJ: Lawrence Erlbaum Associates.

Mehan, H. (1978). Structuring school structure. *Harvard Educational Review*, *48*(1), 32-64.

Nardi, B. (1996). *Context and consciousness: AI and Human Computer Interaction.* Cambridge, MA: MIT Press.

Polman, J. L. (2000). *Designing project-based science: Connecting learners through guided inquiry*. New York: Teachers College Press.

Roschelle, J., Pea, R., Hoadley, C., Gordin, D., & Means, B. (2001). Changing how and what children learn in school with collaborative cognitive technologies. *The Future of Children, 10*(2), pp. 76-101. (Special issue on *Children and Computer Technology*, M. Shields (Ed.), published by the David and Lucille Packard Foundation, Los Altos, CA).

Salomon, G. & Globerson, T. (1989). When teams do not function the way they ought to. *International Journal of Educational Research, 13,* 89-99.

Schwartz, D. L., Lin, X., Brophy, S., & Bransford, J. D. (1999). Toward the development of flexibly adaptive instructional designs. In C. Reigeluth (Ed.), *Instructional Design Theories and Models, Volume II.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Smith, D. C., & Cypher, A. (1998). Making programming easier for children. In A. Druin (Ed.), *The design of children's technology.* San Francisco: Morgan Kaufmann.

Vygotsky, L. S. (1978). *Mind in society: The development of higher order psychological processes.* M.Cole, V. John-Steiner, S. Scribner, & E. Souberman (Eds.). Cambridge, MA: Harvard University Press.

Zuboff, S. (1988). *In the age of the smart machine: The future of work and power.* New York: Basic Books.