Chapter 8

MODELING CSCL SCRIPTS – A REFLECTION ON LEARNING DESIGN APPROACHES

Yongwu Miao, Andreas Harrer, Kay Hoeksema, and Heinz Ulrich Hoppe
Universität Duisburg-Essen

Abstract: The design of collaboration scripts is a new focus of research within the CSCL research community. In order to support the design, communication, analysis, simulation and also the execution of collaboration scripts, a general specification language to describe collaboration scripts is needed. In this chapter, we analyze the suitability and limitations of IMS LD for modeling collaborative learning processes. Based on the analysis, we propose an approach to designing a CSCL scripting language. This chapter presents the conceptual framework of this modeling language and the solutions to the identified problems of IMS LD for formalizing collaboration scripts. Especially, we compare the two approaches through modeling the same collaboration script by using IMS LD and our own CSCL scripting language.

1. INTRODUCTION

According to O'Donnell & Dansereau (1992) a collaboration script is a set of instructions specifying how the group members should interact and collaborate to solve a problem. The term *script* was initially used in schema theory by Schank and Abelson (1977). According to schema theory, a script is a mental structure representing the people's knowledge about actors, objects, and appropriate actions within specific situations. When members of a learning group interact with each other, a shared script can help them to reduce the uncertainty about coordination efforts (Mäkitalo, Weinberger, Häkkinen, & Fischer, 2004), because they know how to behave and what to expect in particular situations. By providing learners with a collaboration script, it is also possible to support learners in aiming at cognitive objectives like fostering understanding or recall (Rummel & Spada, this volume). Additionally, collaboration scripts might also foster the development of metacognitive, motivational, or emotional competence (Kollar, Fischer, & Hesse, in press). A collaboration script is normally represented in the learners' minds (internal representation) and can be represented somewhere in the learning environment (external representation) with complex interplay between these two levels of representation (Carmien, Kollar, Fischer, & Fischer, this volume). King elaborates on the cognitive perspective of CSCL scripts (King, this volume). Because we focus on using collaboration scripts in computer settings, we are interested in representing collaboration scripts in a formal way so that they can be processed by the computer. Such a computational representation of a collaboration script is called a CSCL script.

The conceptual components of a collaboration script and their relations have been discussed in literature (Dillenbourg, 2002; Kollar et al., in press). However, a general modeling language for formalizing collaboration scripts is still missing and most CSCL scripts are embedded or encoded into the learning support environment. Furthermore, there are only few corresponding authoring tools for CSCL practitioners to create, reuse, integrate, and customize CSCL scripts without substantial prerequisites of technical knowledge; there are some proposals for script modeling based on finite automata (Haake & Pfister, this volume) or statecharts (Harrer & Malzahn, 2006) to represent more complex learning processes than linear ones, yet this representation might still be unfamiliar to the educational practitioner. As a first step in the direction of a general CSCL scripting language we investigate in existing learning process modeling languages. The most important attempt in the current discussion in this direction is IMS Learning Design (IMS LD; see IMS LD Website), a standard published by the IMS consortium based on the earlier Educational Modeling language (EML) developed at the Open University of the Netherlands OUNL (Koper, 2001). It is claimed that IMS LD can formally describe any design of teaching-learning processes for a wide range of pedagogical approaches (Koper, 2001; Koper & Olivier, 2004). This modeling language has strengths in specifying personalized learning and asynchronous cooperative learning. However, IMS LD provides insufficient support to model group-based, synchronous collaborative learning activities. Caeiro, Anido, and Llamas (2003) criticized IMS LD regarding CSCL purposes and suggested a modification and extension of the specification. This modification and extension focuses on the elements role-part and method part. Hernandez, Asensio, and Dimitriadis (2004) suggested adding a special type of service, called "groupservice" to extend the capacity of IMS LD. Such an extension at service level, rather than at activity level, cannot appropriately capture the characteristics of collaborative learning activities, because different services may be able to support the same collaborative activity.

The research work presented in this chapter aims at developing a scripting language for formalizing CSCL scripts and exploring their potential types of usage and system support possibilities. In this chapter, first we explain how a scripting language can help CSCL practitioners (e.g., teachers and students) in the design phase (e.g., editing, communicating, predicting, simulating) and in the execution phase (e.g., configuration, monitoring, scaffolding). Then, we clarify the limits of IMS LD when working on a computational methodology for the scripting of collaborative learning processes . Based on the analysis, we propose an approach to design a CSCL scripting language. Rather than a systematic description of the CSCL scripting language, we present it by focusing on how the identified problems of IMS LD for CSCL scripts are solved. In order to compare these two approaches, we present how to model an example collaboration script with IMS LD and by using our CSCL scripting language.

2. POTENTIAL USES AND SYSTEM SUPPORT OF CSCL SCRIPTS

In the following we divide the potential uses of a CSCL modeling language and the computer support it can enable into *usage types during design time* and *usage types while students are performing the learning activities* defined by a designed model. The first category is mainly oriented towards the support of the designer in creating CSCL scripts, while the latter category targets the amount of help a computer system can provide in implementing effective scripts. Dillenbourg and Jermann provide a more general discussion of the added value of computer support for learning scripts in (Dillenbourg & Jermann, this volume).

2.1 Design time uses

The specification of learning processes using a modeling language may have a broad variety of purposes on the part of the designer. Some educational designers use it as a note taking tool for lesson planning. Created models can be saved and used (complete or partially) as a basis for further development. Models can be used for communication between designers. Even at an early state of development, when the model is far from being operational, it can already express educational ideas. Though, due to the complexity of collaborative learning processes, the models get excessively complex and hard to understand. Therefore either reduction of the complexity (by applying projections of specific elements or filtering techniques) or the separation into different perspectives is a typical way to cope with the complexity. The designer can switch between the different perspectives to keep an overview, always choosing the perspective most suitable for further authoring. For learning processes typically the following aspects are relevant and thus candidates for special perspectives:

Procedural/Temporal Perspective. Naturally the sequencing and timing, that is, the process related aspects of the whole learning process, should be represented explicitly

Artifacts Perspective. Artifacts given as resources, used as temporary results and the final outcome of learning activities constitute an important aspect of learning processes. Especially the change of artifacts over time (version history) is information to consider by all participants of a learning process.

Roles Perspective. For organization of specific tasks in group processes the various roles needed for the tasks are an essential information, not only during design time.

Individual/Group Perspective. To get an impression of the workload of one specific member or one subgroup within a group process a perspective stressing these individual aspects is a valuable information for the designer to keep balance between the participants of the process.

The more details of a collaborative learning process are defined, the more the authoring system can provide help to the designer. For example, dependencies or constraints between elements can be highlighted, such as necessity of sequential phases or synchronizing the flow after a split into cooperative sub processes. If the designer specified temporal constraints (minimum or maximum time) for elements of the process, techniques from operations research, such as optimization in network flows or critical path analysis can be applied. A simulated execution of the specified learning process can give the designer a more profound feedback on "what works and what does not?". Imagine the benefit of doing a *simulation run* with information about sequence, time requirements, and produced artifacts before applying the whole design to a real learning situation. The plausibility of the design can be checked much easier than just based on the static structure of the model. Deadlocks (e.g., when subgroups are waiting for each others' input) in the process specification can be detected before making the bitter experience in practical use.

2.2 Runtime uses

The first, weak approach to operationalizing the learning process for the target user "at run time" is the configuration of the learning environment with available tools, resources, communication structure and so on. If this configuration is done once without dynamic addition and removal of elements we call this static configuration. "Compiling and instantiating" such an environment from the specification should be the minimal functionality of a system meant for "playing" the learning design.

While running a learning process model the system can monitor the activities performed by the students. Monitoring functionality could be used twofold: On the one hand the information can be used internally to adapt the process according to the exact specification, on the other hand the monitored information can be visualized to participants of the learning process and give them information on what they have done and produced. This additional feedback can be used to promote reflection about the process or the participants' own behavior, e.g., to stimulate meta-cognitive activities.

At the "informed end" of the spectrum of computer support we see the potential use of the system for scaffolding the learning process, especially when the "typical path" through the process was left by the participants (Koedinger et al., 2004). An enriched specification can give advise to and offer a scaffold to the learners on "what and when to do, how they can play their assigned role best" and so on. Depending on the strictness of the scaffolding the system's behavior can vary between an unrestraining advisor and an intervening tutor. Ideally a script could contain dynamic aspects for adaptive fading in and fading out of scaffolds for the learners.

3. INVESTIGATING THE CAPACITY OF IMS LD FOR FORMALISING COLLABORATIVE LEARNING SCRIPTS

A collaborative learning experience can be described by a collaboration script. Many collaboration scripts have been designed, tested, and even embedded in CSCL applications (e.g., Hoppe & Ploetzner, 1999; Guzdial & Turns, 2000; Miao, Holst, Haake, & Steinmetz, 2000; Pfister & Mühlpfordt, 2002). When using IMS LD to formalize collaboration scripts, we see several major difficulties and challenges:

Modeling groups: Modeling group work with IMS LD raises the problem how to model multiple groups with the same role and how to model the dynamic changes of groups. IMS LD allows for defining multiple roles. Each role can be played by multiple persons. When investigating, we found that in many cases the notational element of "role" can be used to model groups for CSCL scripts. However, by using IMS LD it is very difficult to specify how a group work pattern is assigned to several groups working in parallel and how sub groups can be defined within these groups. If each group or subgroup is defined as a role, the designer has to define a list of roles representing multiple groups. The problem of this solution is that the number of groups in a run is unpredictable during the modeling phase. If only one role is defined for all members of all subgroups, then the information about groups or subgroups will be missing and the run-time system cannot support inter-/intra-group collaboration appropriately. In addition, in IMS LD roles are assigned to persons before running a unit of learning and these assignments stay unchanged within the life cycle of the run. However, in some situations groups are formed and group members are assigned after the start of the process execution. Therefore, in some situations, the notational element of role cannot meet the requirement to model groups.

Modeling artifacts: A second major difficulty while modeling CSCL scripts with IMS LD is the modeling of artifacts. In learning processes, actors usually generate artifacts such as a vote, an answer, an argument, or a design. In IMS LD, an artifact can be modeled as a property, for example a property of a person or a role, that creates the artifact. This property can be used to maintain information such as the learning outcome of a person or a role and to support personalised learning. In collaborative learning processes, an artifact is usually created and shared by a group of people. It is normally used as an object of mediation to facilitate indirect interaction among group members. It may be created in an activity and used in other activities like in an information flow. In order to support group interaction, an artifact should have attributes such as artifact type, status, created_by, creation_activities, contributors, consume_activities, current_users, and so on. By using IMS LD to model an artifact as a property, one has to model all attributes of the artifact as properties as well. These properties should be defined as a property-group with specific constraints. Such a complex definition cannot be understood intuitively. It will be very difficult to model dynamic features even for technically experienced designers, because the limited data-types of properties and the number of references needed make it very complicated to handle artifacts. In addition, it is difficult to model a collective artifact, because IMS LD does not support array-like data-types for a property.

Modeling dynamic features: A third major difficulty while modeling CSCL scripts with IMS LD occurs when modeling dynamic process aspects. IMS LD provides two categories of operations on process elements: read-access operations ("getters") to get the state of process elements (e.g., users-in-role, datetime-activity-started) and write-access operations to change the state of process elements (e.g., change-property-value, hide/show elements, and send notification) to model dynamic features of learning processes. For modeling collaborative learning processes, more of these read and write operations are needed. At least, process element operations concerning our proposed extensions like group and artifact should be extended. In addition, some destructive or constructive operations (e.g., form a group with only male members) should be added. Furthermore, more complicated operations based on these elementary operations will be performed by run-time systems or by users (e.g., to do the configuration and logistics work such as distributing artifacts within a group). Adding such actions will empower learning designers to model complicated processes without being bothered by the technical complexity.

Modeling complicated control flow: A fourth major problem is how to model complex process structures. IMS LD provides play, act, role-part, and activity-structure to model structural relations at different levels. Primarily learning/teaching processes that are structured in a sequential way with concurrently executable activities can be modeled. However, as Caeiro et al. (2003) pointed out, the linear structure of a play with a series of acts introduces a great rigidity while modeling network structures. Although it is possible to model non-linear structural relations among activities by using conditions and notifications, the specification of a collaborative learning process might be very complicated and confusing.

Modeling various forms of social interaction: The last difficulty we want to stress in this chapter occurs when modeling various forms of social interaction. IMS LD uses a metaphor of a theatrical play to model learning/teaching processes. A play consists of a sequence of acts and within an act there is a set of role-parts. These role-parts can run in parallel. Role-parts enable multiple users, playing the same or different roles, to do the same thing or different things concurrently on the same act. For example, while each student reads the same article, the teacher prepares presentation slides. If a group of people performs a synchronous activity, IMS LD enables them to use a conference service and provides no means at the activity level to support collaboration. In collaborative learning processes, it is quite usual that people with the same or different roles perform a shared activity through direct or indirect interaction. While making the joint effort, people with different roles may have different rights to interact with other roles and the environment. In particular, it can not be clearly modeled by using IMS LD whether and how people collaborate, because people may work in a variety of social forms: Individually, in an informal group, in sub-groups, in a group as a whole, or in a community.

4. AN APPROACH TO REPRESENT CSCL SCRIPTS

In order to enhance effective collaboration designs, we have developed a CSCL scripting language to represent collaboration scripts. Because of the limited space of the chapter, we briefly present the CSCL scripting language by explaining the core concepts and their relations, rather than giving a systematic description. Then we focus on describing how the identified problems of IMS LD for CSCL scripts are solved in our scripting language by introducing the required constructs on the conceptual level. This does not necessarily imply that we want to provide a completely independent approach for formalizing learning processes, including providing our own interpreting machine or engine. At the moment we are still in the process of exploring if the existing standard can be extended according to the identified needs. Another possibility is to consider our approach as a higher-level one closer to the practitioner's and researcher's needs that can be "compiled", that is, semantically mapped to the existing description format. This question in its completeness is unresolved, but we will give some details on the aspects that we consider to be resolved at the moment.

4.1 A conceptual basis for CSCL scripting

In this subsection, we briefly present the core concepts and their relations of the CSCL scripting language.

A CSCL script is a specific learning design which emphasizes collaboration. A CSCL script contains contextual information that applies to other elements within the process. As shown in Figure 8-1, a CSCL script consists of a set of roles, activities, transitions, artifacts, and environments. A CSCL script has attributes such as learning objectives, prerequisites, design rationale, coercion degree, granularity, duration, target audience, learning context, script specific properties, and generic information (e.g., id, name, description, status, creation date, and so on). The attribute *design rationale* enables to express and communicate the design ideas and underlying pedagogic principles. The values of the attribute *coercion degree* represent different degrees of *informedness*. CSCL scripts with different coercion degrees have different usages, which will be discussed later in the chapter. If a CSCL script of fine granularity is embedded in a CSCL script of coarse granularity, the mappings between the roles, properties, and artifacts of two CSCL scripts should be specified. A role is used to distinguish users who have different privileges and obligations in the processes described in the CSCL script. Both persons and groups can take a role. A group can have subgroups and person members. An activity is a definition of one logical unit of a task performed individually or collaboratively. There are three types of activities: atomic activity, compound activity, and route activity. A compound activity is decomposable into a set of networked activities and even other scripts. A transition specifies a relation of temporal dependency between two activities. An artifact may be created and shared in and/or across activities as an intermediate product or a final outcome or both. An environment can contain sub-environments and may contain tools and contents. A tool may use artifacts as input parameters or output parameters or both. A content is a kind of learning object which exists and is accessible. An action is an operation and may be performed by users during an activity or by the system before or after an activity. A property may be atomic or may have internal structure. An expression may use properties and other expressions as operands. Like IMS LD, a condition refers to a condition clause which is defined as an if-then-else rule consisting of a logical expression and actions, transitions, and/or other conditions. Actions, properties, expressions, and conditions have very complicated relations with other process elements (e.g., scripts, roles, activities, artifacts, persons, groups, environments, and so on). For example, an action may use process elements as parameters and change the values of attributes of certain process elements. Such relations are not drawn in this diagram in order to keep the diagram simple and readable.

Using the scripting language to formalise a collaboration script means specifying how persons or groups or both, playing certain roles, work collaboratively towards certain outcomes (which can be artifacts) by performing temporally structured activities within environments, where needed tools and content are available. Actions, properties, expressions, and conditions are useful to model more complicated, dynamic control-flow and information flow in collaborative learning processes.

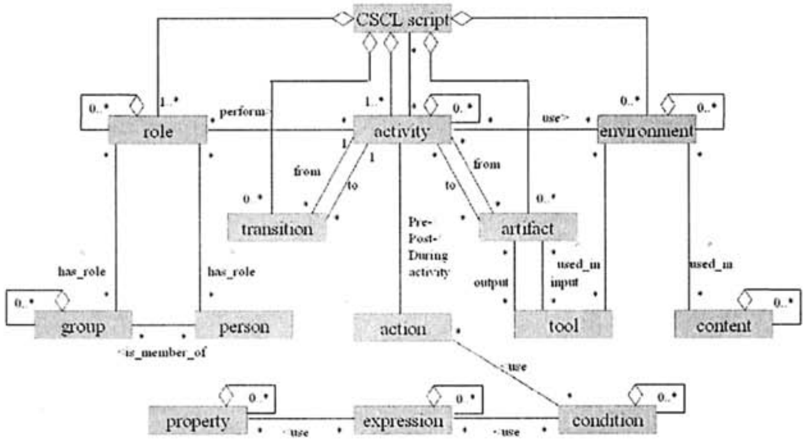


Figure 8-1: Core modeling elements and their interrelation

4.2 Solutions

In this subsection, we focus on presenting our solutions to the identified problems of IMS LD for CSCL scripts.

Explicitly introducing groups. The introduction of a group element enables us to model group based collaboration in a simpler and more intuitive way. In our CSCL scripting language, a group is modeled by using attributes such as name, max-size, min-size, person members, super-groups, subgroups, engaged roles, form-policy, disband-policy, dynamic/static, and runtime information. In addition, local-/global group properties are added for learning designers to define additional attributes of a group. One or more groups can play the same a role. Therefore, when a role is defined and is assigned to carry out an activity, it does not matter how many groups will play this role at runtime. On the one hand, a group can have subgroups and form a hierarchically structured organization (a directed-acyclic-graph). Any change in the organization has no effect on the definition of the role in scripts. On the other hand, re-definition of roles in scripts does not effect organization. This proposal raises the question when to model a group and when to use a role for a group. From our perspective, some roles are organization oriented definitions like students and staff. Others are behavior-oriented roles such as meeting chairman and tutor. It would be better to model an organization-oriented role as a group role and to model a behavior-oriented role as a role for assigning tasks.

Explicitly introducing artifacts. The artifact element does not exist in the IMS LD specification. As we explained already, the usage of artifact elements can enable to model CSCL contexts much more intuitive and easier than to model the same process within IMS LD, because some burdens on the designers to handle technical tasks are avoided by providing built-in mechanisms. In our language, an artifact is treated as a file which can be a MIME-type or user-defined type. The attributes of an artifact contain generic information (e.g., title, description, type, status, URL, sharable, and aggregated), association information (e.g., creation_activities, consume_activities, and default_tool), and run-time information (e.g., created_by, creation_time, contributors, last_modification_time, current_users, locked_status, and so on). An artifact and its status will be accessible in the environment of the creation-/consume- activities at run-time. The specification of the relations between artifacts and tools will help the run-time system to pass artifacts as input/output parameters to and from tools automatically at runtime. Some expressions and actions related to artifacts should be added for mediating group work such as get-current-users-of-artifact and change-artifact-status. The artifact-specific properties may be useful to model a specific feature of an artifact. As an aggregated artifact, it is possible to append collective information to the same file.

Extending actions and expressions. An action is a generic and powerful mechanism to model dynamic features of a collaborative learning process. We add some actions as components of the CSCL scripting language that can be executed directly by the runtime system. In addition, we add an action declaration mechanism for experts to define a procedure by using the CSCL scripting language. In order to support the definition of complicated procedures, we add a "collection" data type and a loop control structure. The defined procedure can be interpreted by the run-time system as process element operations, and in turn, as executable code. Therefore, complicated actions can be defined by using an action declaration and assigning the parameters needed. IMS LD provides a limited set of actions such as property operations, showing/hiding entity, and notification. The action notation we introduced provides a unified form of operations including not only actions defined in IMS LD but also commonly used operations concerning script, activity, artifact, role, group, person, transition, environment, and their relations. An expression is defined as it is in IMS LD: some read operations can be used as operands in expressions like "is-member-of-role", "datetime-activity-started", and "complete". However, it is necessary to add read operations to support collaboration such as "are-all-role-members-online" and "artifact-contributors". Furthermore, corresponding to the action declaration, we add an expression declaration mechanism for experts to define complicated expressions which could be used by normal teachers and students.

Introducing transitions and routing activities. We partially accept the suggestion of Caeiro et al. (2003) to introduce transitions and routing constructs recommended by the Workflow Management Coalition (WfMC Website). Because interactions of person-to-person, group-to-group, and role-to-role and splitting and synchronization of process threads are never restricted at higher levels, we have to use such a mechanism not only at play level but at all possible levels in order to model the arbitrarily complicated structural relations among activities.

Using activity-centered methods to assign roles. We give up the metaphor of a theatrical play and the role-part method. Instead, we use an activity centered role assignment method. In the CSCL scripting language, for modeling an activity, the attributes are defined to specify engaged roles, used environments, input/output artifacts, transitions and restrictions, pre-/post-/during activity actions, user-defined activity-specific properties, completion-mode, execution-time, completion-condition, mode of interaction, social plane, interaction rules, generic information, and simulation information. Some attributes are important for designers to model collaborative processes and some for the run-time system to configure collaborative learning environments appropriately for users. For example, the possible values of social planes are: separately with a certain role, individually with a certain role, collaboratively with one or multiple roles or both, collaboratively in sub-groups with a certain role, and so on. If the choice is "separately", the run-time system will create an activity instance for each user starting the activity. If anyone completes his activity, all activity instances terminate. "Individually" means that the run-time system will create an activity instance for each user. The run-time system synchronizes access to the following activity by continuously checking whether all users have already completed the current activity. In comparison, the run-time system based on IMS LD typically handles this situation defined by using the role-part method. The choice of "collaboratively with one and/or multiple roles" makes the run-time system create only one activity instance and a session facilitating collaboration. The semantics of the value "collaboratively in subgroups with a certain role" is that the run-time system creates an activity instance and a session for each sub-group and the members of each sub-group can have a shared activity workspace. The run-time system synchronizes access to the next activity when all subgroups finish their work. Another example is the attribute *interaction rules*. An interaction rule specifies under which condition which role can (not) perform which actions. For example, the tutor can perform the actions to create (sub)groups and assign group members. Such information can be used by the run-time system to automatically provide corresponding awareness information in the user interface to help users to perform specified actions. In short, interaction rules explicitly specify different responsibilities of different roles in a collaborative learning activity.

5. MODELING A COLLABORATION SCRIPT WITH IMS LD AND THE CSCL SCRIPTING LANGUAGE

In this section, a collaboration script is used as an example. We discuss how this collaboration script can be modeled by using IMS LD and by using our CSCL scripting language. Our example will be the "Knowledge Convergence Script" (Weinberger, Fischer, & Mandl, 2004, and Weinberger, Stegmann, Fischer, & Mandl, this volume), that has been shown to be effective in improving the learners' convergence either on epistemic or on the social level.

In short this script consists of the following phases and interactions between the members of groups of three students:

- Phase 1 - case reporting: Each student gets information about a (educational) case and is writing a report about the case.

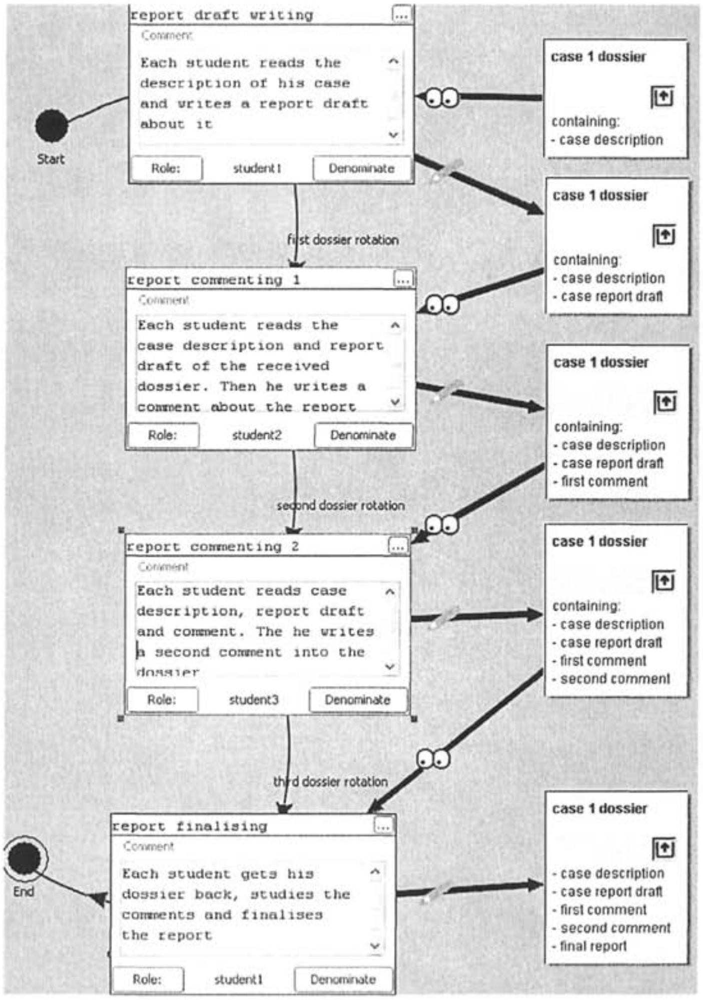
- Phase 2 - criticizing 1: Each student gets the case and the report of the student to his left and writes a comment about the report.
- Phase 3 - criticizing 2: Each student gets a case, the report and the comment the student to his left produced in phase 2 and writes a second comment about the report.
- Phase 4 - Finalizing the report: Each student gets back his own report together with the comments of the two other students and rewrites it taking the comments into account.

The flow of the artifacts produced by the students, specifically the artifacts in relation to Case 1, can be seen in the graphical schema in Figure 8-2.

5.1 How to model the script by using IMS LD

IMS LD is designed mainly for supporting web-based learning environments and the run-time environment will render the web pages for users according to the definition of the unit of learning. To give an impression of the design work we will abstract from generation of HTML and XML content pages, but focus on the major steps in the design process for the Knowledge Convergence Script:

1. Define three roles for the three group members, since IMS LD does not explicitly represent groups. Each role will be constrained to have at most 1 person playing the role.
2. Define 12 properties for the reports and comments produced by the students, because each student writes a report, two comments on the others' reports, and a final version of the report. Properties are the means of choice in IMS LD, because they can be flexibly used for person- or role-related aspects, thus also as a substitute for a missing "document/artifact" construct. For a better structuring it is advisable to compose sets of properties, such as all documents related to Case 1, in so called property-groups, that contain references to their constituents.
3. Define the 12 activities that the learners should perform in this script and their effects on the properties representing the documents (i.e., the products of student writing). These properties have to be set explicitly from the outside, that is, from an external service or from a learning object document.
4. Predefine the document flow (represented in the properties) for each step of the script explicitly, such as "Student 1 has to get report 3 from student 3, Student 2...". This is statically defined for a fixed number of documents and learners.



```
Each student reads the
description of his case
and writes a report draft
about it
```

```
Each student reads the
case description and report
draft of the received
dossier. Then he writes a
comment about the report
```

```
Each student reads case
description, report draft
and comment. The he writes
a second comment into the
dossier
```

```
Each student gets his
dossier back, studies the
comments and finalises
the report
```

Figure 8-2: Diagram showing the flow of the dossier of Case 1 through activities

5.2 How to model the script by using the scripting language

The same process will now be sketched for the CSCL script representation presented in the previous sections. Our main focus is also on the general overview with some details about practical and technical issues of applying and implementing this notation:

1. Define a group of three members explicitly; there is still also the option of defining groups by roles, but the notation also offers a dedicated "group" construct to the designers.
2. Explicitly define three artifacts that represent the documents produced by the students. For better structuring these artifacts can be aggregated to "composite artifacts" (e.g., one dossier for all documents related to one case and even a collection of all dossiers) of complex structure.
3. Define actions for the initial distribution and the re-assignment/rotation of artifacts to the group members. The independence of concrete numbers for documents and persons is highly desirable, so that the action can be re-used in different situations or stages of the learning process. These actions can be freely defined by a learning designer, if he has some understanding of specifying procedures on an abstract level. In our example the two actions "DistributeArtifactCollection" and "RotateArtifactCyclic" would be very useful, especially the latter, because it is performed after every writing phase of the students, but with different actors getting the dossiers. To give an impression of the specification level of such a generic action we give some pseudo-code representation for "RotateArtifactCyclic" and a graphical schema for this procedure (see Figure 8-3), that gives the dossier to the next group member in sequence.

```
rotateArtifactCyclic(ArtifactCollection art, Group learners){
  while (art.hasMoreElements()){
    assign(art.currentElement(), group.nextMember());
  }
  assign(art.lastElement(), group.firstMember());
}
```

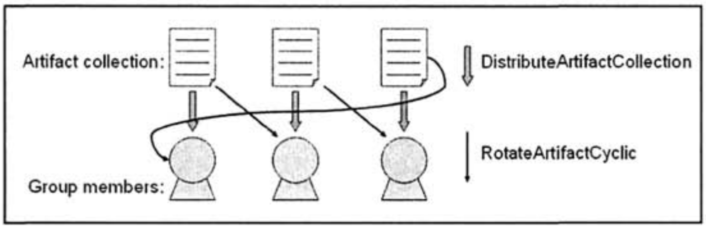


Figure 8-3: The graphical schema for explaining two actions: "DistributeArtifactCollection" and "RotateArtifactCyclic"

4. Use the action to rotate artifacts in the learning design for each step of the script; the advantage of having a generic action definition is now that the action can be re-used now by calling this action with different parameters (the current states of the dossiers and the group) without any manual assignment of the respective documents. This re-use can be done for self-defined actions and also for any library of pre-defined actions that other learning designers created. Thus, in case that a suitable pre-defined action is already available (such as the mentioned "RotateArtifactCyclic" that we defined for our own purposes), the Step 3 can be skipped, which is especially desirable for practitioners without programming skills. Pre-defined actions can be used conveniently in our tree-based editor tool, by choosing and parameterizing the appropriate actions from a list (see Figure 8-4).

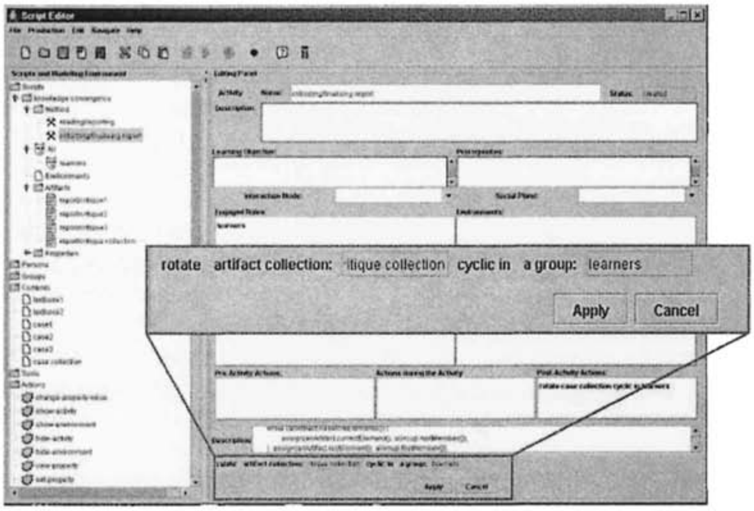

rotate artifact collection: itique collection cyclic in a group: learners

Figure 8-4: Define a post-activity action by assigning parameters

Figure 8-4 shows a screenshot of our tree-based authoring tool when defining the script. The left panel is used to define the script elements (two activities, a group, three artifacts and an artifact collection) and their structural relations. The right panel is used to create a detailed design for each process element, currently for the "criticizing/finalizing-report" activity. The enlarged part illustrates how a post-activity action can be defined in a user-friendly manner. Users can assign the parameters of the action by dragging an element node from the structural tree in the left panel and dropping into the parameter boxes of the action representation.

5.3 Comparison of the two approaches

Although the example script does not cover all features that we discussed in the chapter, we can see the differences when modeling the script with IMS LD and with the CSCL scripting language. We hope that we can provide added value in different respect:

First, the use of a conceptual level, that is closer to the concepts practitioners use, such as the availability of explicit group definitions and artifacts produced by the participants, enables a better understanding for the designer and also in the discussion between practitioners than the IMS LD constructs, such as properties and roles as substitute for groups, offer.

Second, the presented approach of defining own actions in a potentially very generalizable way (using parameters), makes these actions much more re-usable than the IMS LD solution where the definition has to be predefined in a static way; the activity "rotateArtifactsCyclic" in our example can be re-used flexibly within the same script or in a completely different one just by using different parameters for both artifact collection and group, while in the LD solution each step has to be edited again; this is especially useful with different numbers of artifacts to distribute to an arbitrary group (which would not be a problem for our generic activity definition).

Our approach has been prototypically implemented in different tools for editing of CSCL scripts. These tools have been presented in more detail in Miao, Hoeksema, Hoppe, and Harrer (2005).

6. CONCLUSIONS

In this chapter we have identified five major limitations of IMS LD when formalizing CSCL scripts. Based on this, we have suggested a scripting language for CSCL. The identified problems of IMS LD are solved in the language respectively by 1) explicitly introducing the group entity to facilitate modeling organizational role and behavior role; 2) explicitly introducing the artifact entity to enable designers to model artifact and information flow easily and intuitively; 3) extending process element operations and providing declaration mechanisms to capture dynamic features of collaborative learning processes; 4) exploiting WfMS routing technologies to enable the specification of complicated control flow; and 5) giving up the metaphor of theatrical play and the role-part and using an activity-centered definition method to model various forms of social interaction. In addition, we briefly discussed the potential usages of CSCL scripts and possibilities of system support.

Through comparing the two approaches of modeling the same collaboration script with IMS LD and with the CSCL scripting language, we see, at minimum, two advantages of our approach: First, at conceptual level, practitioners can use terms that are closer to the concepts they use in practice. It will be helpful for them to understand and design teaching/learning process models. Second, using actions in our approach makes it possible for practitioners to model complicated processes, because the burden of practitioners to handle technical complexities is reduced.

REFERENCES

Caeiro, M., Anido, L., & Llamas, M. (2003). A critical analysis of IMS learning design. In B. Wasson, R. Baggetun, U. Hoppe, & S. Ludvigsen (Eds.), *Proceedings of the International Conference on Computer Support for Collaborative Learning - CSCL 2003, COMMUNITY EVENTS - Communication and Interaction* (pp. 363-367). Bergen, NO: InterMedia.

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

Guzdial, M., & Turns, J. (2000). Effective discussion through a computer-mediated anchored forum. *Journal of the Learning Sciences, 9*(4), 437-469.

Harrer, A., & Malzahn, N. (2006). Bridging the Gap - Towards a Graphical Modeling Language for Learning Designs and Collaboration Scripts of Various Granularities. *Proceedings of International Conference on Advanced Learning Technologies (ICALT 2006).* Los Alamitos, CA., IEEE Press.

Hernandez, D., Asensio, J.I., & Dimitriadis, Y. (2004). IMS Learning Design Support for the Formalisation of Collaborative Learning Flow Patterns. *Proceedings of the 4th International Conference on Advanced Learning Technologies (Aug.30 - Sep. 1, 2004),* (pp.350-354). Joensuu, Finland: IEEE Press.

Hoppe, U.H., & Ploetzner, R. (1999). Can analytic models support learning in groups. In P. Dillenbourg (Ed.), *Collaborative-learning: Cognitive and Computational Approaches* (pp.147-168). Oxford: Elsevier.

IMS LD Website. Retrieved August 05, 2005, from http://www.imsglobal.org/learningdesign/index.cfm

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. M., & Hockenberry, M. (2004) Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. In the *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS-2004),* Maceio, Brazil.

Kollar, I., Fischer, F., & Hesse, F. W. (in press). Collaboration scripts – a conceptual analysis. *Educational Psychology Review.*

Koper, E. J. R. (2001). *Modeling units of study from a pedagogical perspective: The pedagogical meta-model behind EML.* Document prepared for the IMS Learning Design Working Group. Heerlen: Open Universiteit Nederland.

Koper, E. J. R., & Olivier. B. (2004). Representing the learning design of units of learning. *Educational Technology & Society, 7*(3), 97-111.

Mäkitalo, K., Weinberger, A., Häkkinen, P., & Fischer, F. (2004). Uncertainty-reducing cooperation scripts in online learning environments. In P. Gerjets, P. A. Kirschner, J. Elen, & R. Joiner (Eds.), *Proceedings of first joint meeting of the EARLI SIGs "Instructional Design" and "Learning and Instruction with Computers"*. Tübingen: Knowledge Media Research Center.

Miao, Y., Hoeksema, K., Hoppe, U., & Harrer, A. (2005). CSCL Scripts: Modeling features and potential use. In T. Koschmann, D. Suthers, & T. -W. Chan (Eds.), *Computer Supported Collaborative Learning 2005: The Next 10 Years* (pp. 423-432). Mahwah, NJ: Lawrence Erlbaum Associates.

Miao, Y., Holst, S.L., Haake, J.M., & Steinmetz, R. (2000). PBL-Protocols: Guiding and controlling problem based learning processes in virtual learning environments. In B. Fishman & S. O'Connor-Divelbiss (Eds.), *Fourth International Conference of the Learning Sciences* (pp. 232-237). Mahwah, NJ: Lawrence Erlbaum Associates.

O'Donnell, A. M., & Dansereau, D. F. (1992). Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. In R. Hertz-Lazarowitz & N. Miller (Eds.), *Interaction in Cooperative Groups: The theoretical Anatomy of Group Learning* (pp. 120-141). London: Cambridge University Press.

Pfister, H.-R., & Mühlpfordt, M. (2002). Supporting discourse in a synchronous learning environment: The learning protocol approach. In G. Stahl (Ed.), *Proceedings of the CSCL2002 Conference on Computer Supported Collaborative Learning, Boulder, USA* (pp. 581-589). Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates.

WfMC Website. Retrieved August 05, 2005, from http://www.wfmc.org/index.html

Weinberger, A., Fischer, F., & Mandl, H. (2004, April). Knowledge convergence in computer-mediated learning environments: Effects of collaboration scripts. *85th Annual Meeting of the American Educational Research Association (AERA)*. San Diego, CA, USA.