

# Extending a Learning Design Editor with a Monitoring Component

Nils Malzahn, Marcel Pokrandt, H. Ulrich Hoppe  
*Collide Research Group, University Duisburg-Essen, Germany*  
malzahn@collide.info

**Abstract:** The MoCoLADe learning process modeling environment has been extended with a monitoring component that facilitates teacher supervision and, as a next step process-adaptive scaffolding. A major challenge in implementing and extending MoCoLADe has been the strive for compatibility with IMS LD as a de facto standard. Practically, IMS LD scripts are exported from MoCoLADe and executed with the Coppercore Runtime Environment for monitoring. Workarounds are necessary to compensate for representational deficits of IMS LD.

**Keywords:** Educational modeling, monitoring, IMS LD, Learning flow

## 1. Introduction

“Learning Design” (LD) denotes the modeling and specification of learning processes, including learning and teaching activities, roles and tools. The de facto standard in this area is IMS LD (see <http://www.imsglobal.org/learningdesign>; [1]). “Educational modeling languages” like IMS LD can be used to specify and potentially operationalize the orchestration of various learning scenarios. Applications of IMS LD are as yet mainly confined to web based learning environments. In [2] the MoCoLADe editor has been presented, which enables the user (teachers, learning designers) to conceptualize, plan and analyze learning processes. This view includes, e.g., the traditional task of lesson planning as part of a teacher’s professional activity, which may be useful even in non-computerized target scenarios. Exploiting the fact that we have a computerized representation, animated walk-throughs and simulations have been added. Although MoCoLADe uses its own representation and processing model, it also provides mappings to IMS LD in the form of an export function. These interactive computational techniques may allow for finding and analyzing problematic parts of the learning design before the actual lesson is enacted.

With respect to the list of desired LD-based functions[3]: (1) computer-based visual editing and re-use, (2) modeling with various perspectives, (3) model-based predictions, (4) simulation of visual models, and (5) model-based scaffolding, MoCoLADe has already served the purposes of (1) to (4) [2]. With the herein presented monitoring approach, MoCoLADe is prepared for supporting scaffolding (5). This the next step towards a round trip learning process engineering.

## 2. Prerequisites

The graphical notation of MoCoLADe is inspired by the semantics of statechart diagrams [4] allowing a formal modeling of collaboration scripts based on the conceptual

framework of Kobbe et al. [5]. IMS LD has some known drawbacks [6] with respect to modeling collaborative learning settings. These include the lack of explicit group building constructs and the absence of expressions for basic loops. So compromises have to be made when exporting models of collaborative scenarios from the MoCoLADE editor to IMS LD.

IMS LD comes with the “Coppercore Engine” as the reference implementation for runtime environments. Coppercore requires a J2EE compliant application server and is bundled with a simple web player, which can be used to execute/display basic learning activities. Furthermore the “Command Line Interface to Coppercore” (Clicc) is part of the Coppercore bundle. This client is used for user registration, role assignment and assignment of users to an available learning design script. The most important functionality of Coppercore in the context of this paper is its capability of interpreting IMS LD scripts and providing the current state of the script, i. e. which user is in which activity, which roles a user has etc.

### 3. Approach

A monitoring function can serve two different purposes: On the one hand the information can be used internally to adapt the process according to the specification. On the other hand the monitored information can be visualized to the students and provide information about what they have done and produced. This feedback can be used to promote reflection about the process or the participants’ behavior. Thus to be able to follow the student’s way through the script the current state of the Coppercore Engine has to be accessed and analyzed. The Coppercore engine organizes its state (basically) into two different data structures: the activity tree and the environment tree. An *activity tree* is a personalized XML representation (s. fig. 1; left), which contains the learning activities to be performed by a user (cf. [7]). . This includes those activities that have already been finished as well as those to come. From this tree the current state of the script is derived. An *environment tree* (s. fig. 1; right) is an XML representation of all learning object and services that belong to a learning activity (cf. [7]).

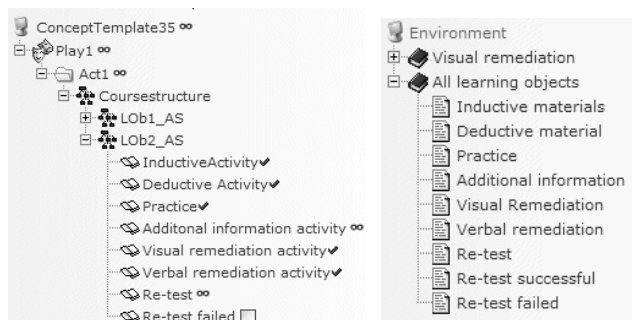


Figure 1: Example of a rendered activity tree and an environment tree [7]

Unfortunately it is not possible to query parts of the activity tree, so that the full tree has to be analyzed every time an update of the monitoring tool shall take place. An update is needed in small, regular intervals or to be always received when a change occurs to the tree happened to enable timely actions by the teacher.

To enable further analysis of the learning design as well as the behavior of the students the movement of all participants shall be logged. This log enables the teacher to comprehend the complete path through the learning script of the particular participants, thus providing opportunities for specific support of the learners or for analyzing the learning process itself. Furthermore the teacher should be enabled to cope with typical problems, i.e.

idle students, missing students, ill-assignment of learning resources. To satisfy these goals together with the student tracking several issues have to be tackled.

### 3.1 IMS LD deficits and workarounds

As there is no specific “grouping construct” in IMS LD, groups have to be represented by other means. The MoCoLADe editor uses explicit grouping in its own representation, but the export has to make use of special roles as proposed in [6, 7] when exporting to IMS LD. While this is a good work-around on the specification level, it has a flaw on the execution level. It is not guaranteed that all the users holding this role advance as an atomic unit through the script. This means that supposedly stable “atomic” groups of users can be distributed over several activities of the learning script. While this cannot be helped, because of the limits of IMS LD, it is a fact that is surely worth to be shown in the monitoring component to enable appropriate actions by the teacher (see fig. 2).

In collaborative learning scenarios it is typical to work on common resources or to distribute resources (e.g. papers on specific topics). It is important to know who owns which resource to get an idea if the script is running correctly. Since IMS LD binds resources to environments, which are bound to whole activities, it is not easy to distinguish who owns a particular resource. This is even worse if the resource is a group resource. The proposed solution is to assume that group resources (i.e. resources owned by the whole group) are those resources that are visible to all group members at the same time. If a resource is only visible to some of the group members, the resource is considered as personally owned. To detect this, the content of the environments has to be investigated and compared. A special meta-data entry in the editor’s IMS LD exported provides a mapping between resource names in the model and the resource ids of Coppercore to ease the representation in the monitoring component.

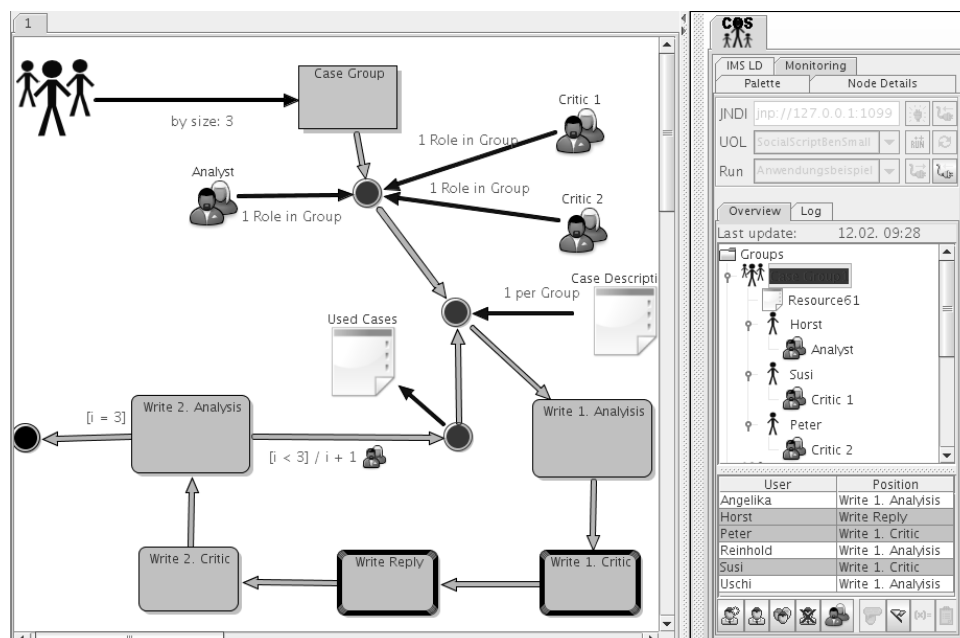


Figure 2: Distribution of group members over several activities in the Social Script.

### 3.2 Monitoring & intervention mechanism

First of all, there is no way of generating the original script model in the editor from a Coppercore activity tree. This is due to the limitations of IMS LD, e.g. concerning loops. This means a repetition of activities may have either been modeled as a loop or as sequence of activities. To solve this issue a unique id identifying the model from which the current instance of the script was generated, is used. Thus it is possible to map the activities of Coppercore to the activity nodes in the editor.

The basic idea of the intervention mechanism is to keep the script running, even though deviations may occur (fault tolerance). That means, it is necessary to handle unforeseen situations like missing or idle students as well as technical problems or erroneous scripts. Therefore the teacher should be provided with an interface which allows to finish an activity for a particular user. Since the sequencing of activities may be controlled by values of properties, it should be possible to adapt these properties to resolve deadlock situations or to be able to adapt a running script to a reduced number of students. Additionally the teacher should be allowed to add and create new Coppercore users to add latecomers to scripts. Furthermore the teacher should be able to contact students directly to stimulate their activity.

The presented approach was implemented by us using Coppercore as a LD execution environment, JMS as a message passing mechanism between Coppercore and the monitoring environment as well as between the monitoring environment and the students, and JBoss as the server running Coppercore and JMS.

## 4 Examples & Discussion

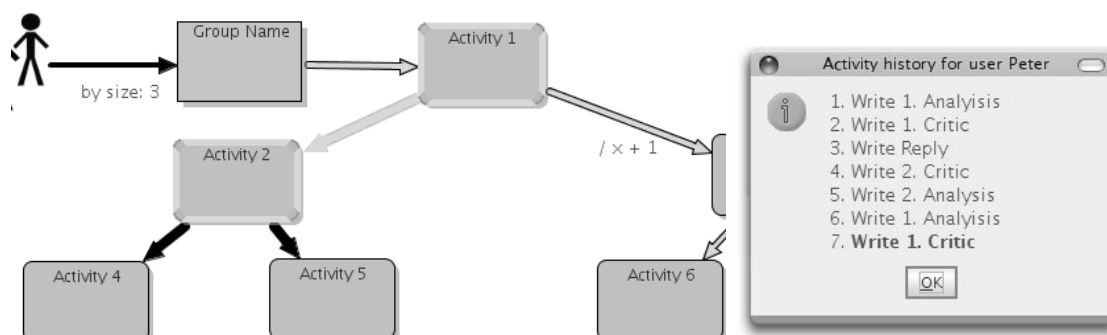


Figure 3: left: Example Model focusing on activity choices; right: Activity History view

Figure 2 and figure 3 show two scripts, which are modeled with MoCoLADe. Figure 3 (right) is just for illustrative purposes. It shows that the way through a script may be monitored by the teacher, because finished activities are highlighted. Finished activities are indicated by a green border (light gray) with curved edges. The red (black) arrows below Activity 2 indicate that the currently monitored user can choose between Activity 4 and Activity 5. Since both activities are marked visible to the student (due to limitations of IMS LD / Coppercore) it is only possible to determine the chosen activity after it has been finished. If there is only one possible outgoing sequencing edge (either by constraints or if there is exactly one) from one activity to another, the monitoring component will assume that the student is working on this activity.

Figure 2 shows the so called “Social Script” (cf. [8]). Within this collaboration script groups of three are assigned to the task of writing analyses and comment on them. Every group member should hold each role (Analyst, Critic 1, Critic 2) before the group finishes the script. Scripts like the social script are repetitive, i.e. they may contain loops. That

means even if it is clear in which activity the student is in, it is not necessarily clear in which iteration of the script. A history view like the one shown in figure 3 (right) provides the necessary information. For the Social Script, it is quite important that the whole group advances from one activity to the next one. Nevertheless the red (black) bar in the right part of figure 2 indicates that one of the groups is spread over more than one activity. If the group is selected for further inspection like it is done in figure 2, those activities where members of the respective group are active are shown with a red (black) border. Depending on the cause of the group division the teacher may now intervene. He or she could either send a message to the idle students or finish an activity for selected users using one of the intervention buttons, which are shown in figure 2 at the bottom right corner of the screen.

## 5 Outlook

So far we documented our effort to extend our graphical learning process modeling and monitoring tool. Due to the limitations of IMS LD and Coppercore, we faced some issues that forced us to compromise, e.g. with respect to group monitoring or resource distribution. While this combination integrates nicely with our previous work on controlling rich learning environments like FreeStyler and CoLab with IMS LD scripts [9, 10], we now plan to transfer the approach to other (online) learning environments. The next important step towards a LD modeling environment fulfilling all of the five desired properties postulated in [2] will be to integrate means for the specification of automatic scaffolding into MoCoLADe. While there is already a basic option to specify scaffolds, there is currently no implementation for exporting these scaffolds into an external representation.

## References

- [1] Kollar, I., F. Fischer, and J.D. Slotta (2005). Internal and External Collaboration Scripts in Web-based Science Learning at Schools. in *Computer-Supported Collaborative Learning 2005*. Mahwah, NJ.: Lawrence Erlbaum Associates.
- [2] Harrer, A., Malzahn, N., & Hoppe H.U. (2007). Graphical Modeling and Simulation of Learning Designs. In T. Hirashima, H. U. Hoppe & S. S. Young (Eds) *Supporting Learning Flow through Integrative Technologies* pp. 291-294, Amsterdam NL, IOS Press.
- [3] Harrer, A. & Hoppe, H. U. (2008). Visual Modeling of Collaborative Learning Processes – Uses, Desired Properties and Approaches. In: Botturi, L., Todd S. (eds) *Handbook of Visual Languages for Instructional Design: Theory and Practices*, 281-298, Information Science Reference, Hershey.
- [4] Harel, D. & Naamad, A. (1996). The state space semantics of statecharts. *ACM Transactions on Software Engineering and Methodology* 5(4), 293-333.
- [5] Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A. & Häkkinen, P. (2007). Specifying Computer-Supported Collaboration Scripts. *International Journal of Computer-Supported Collaborative Learning*, 2(2-3), New York, Springer.
- [6] Miao, Y., Hoeksema K., Hoppe, H. U. & Harrer, A. (2005). Scripting Languages for Collaborative Learning: Modelling Features and Potential Use. *Proc. of CSCL 2005*, Taipei, Taiwan.
- [7] Vogten, H. (2006). CopperCore: a service based approach towards implementing the IMS Learning Design specification. Available from <http://hdl.handle.net/1820/707> [accessed 26/05/2008].
- [8] Weinberger, A. (2003). Scripts of Computer-Supported Collaborative Learning: Effects of social and epistemic cooperation scripts on collaborative knowledge construction. PhD Thesis, LMU München.
- [9] Harrer, A., Lucarz, A., & Malzahn, N. (2007). Dynamic and Flexible Learning in Distributed and Collaborative Scenarios Using Grid Technologies. Groupware: Design, Implementation, and Use, *Proc. of CRIWG 2007*, LNCS 4715:239-246.
- [10] Harrer, A., Malzahn, N., & Roth, B. (2006). The remote control approach - how to apply scaffolds to existing collaborative learning environments. *Proc. of CRIWG 2006* pp. 118-131.