Harrer, A., Malzahn, N., Hoeksema, K. & Hoppe, U. (2005).
Learning Design Engines as Remote Control to Learning Support Environments.
Journal of Interactive Media in Education 2005(05).
[jime.open.ac.uk/2005/05].

25 August 2005

ISSN: 1365-893X

Learning Design Engines as Remote Control to Learning Support Environments

Andreas Harrer, Nils Malzahn, Kay Hoeksema, Ulrich Hoppe

Abstract:

Context:

Chapter 5 of the Learning Design book describes the operational model of a learning design engine based on the concept of finite automata with output alphabet. We rely on this event concept to include pre-existing learning tools in flexibe and rich learning designs.

Contribution:

We sketch an approach for the integration of complex learning environments in learning designs. Interactive learning support environments, such as argumentation or modelling tools are pre-existent and have a high potential when integrated in learning designs.

We propose an approach that aims at a clear separation of the learning design engine, the specification of the learning flow (as LD documents) and learning environments. According to its current state, the engine controls the learning environment with events (such as "start a new phase"), defined as a vocabulary for a set of environments, that are mapped to the environments' existing functionality (such as "create new workspace"). Thus the engine remotely controls the learning tools while the tools can initiate state transitions in the engine on specific events in the tool.

Keywords: Learning Design Engine, Learning Support Environment, Interoperability

Commentaries:

All JIME articles are published with links to a commentaries area, which includes part of the article's original review debate. Readers are invited to make use of this resource, and to add their own commentaries. The authors, reviewers, and anyone else who has 'subscribed' to this article via the website will receive e-mail copies of your postings.

1 Learning Support Environments without Learning Design Engines and vice versa

Up to now complex learning support environments (LSE) and learning design plays are largely unrelated and co-exist, but do not co-operate. On the one hand learning support environments, such as WISE (2005), Co-Lab (2005) or Belvedere (Suthers, Weiner, Connelly, & Paolucci (1995)), either have a specific ("hard-wired") process model embedded or do not have an explicit learning process model at all. On the other hand environments that use learning design documents as process scaffolds or "scripts" are usually oriented towards delivery of web-content and some simple services, such as conference tools. Making the learning processes explicit in a formal specification, such as IMS/LD, offers the possibility to re-use the pedagogical rationale that is reflected within the specification. This makes it attractive for pre-existing learning support environments to utilize the formal character of IMS/LD and the availability of learning design engines (LDE), such as CopperCore (2005), to have an explicit process support within the learning environments without having to implement a process model from scratch for each individual environment. In the next section we will present our approach to achieve synergy between both lines of computer-based learning and an architecture supporting this approach.

2 Bringing it all together – flexible integration of LSE and LDE

Our starting point that leads to an integration effort of these both lines of development and research in computer-supported learning was, that we have been using the collaborative modelling and discussion support environments Cool Modes (Pinkwart, 2003) and FreeStyler (Hoppe & Gaßner, 2002) in a wide variety of domains (such as brainstorming, UML modelling, System Dynamics modelling, creation of argumentation graphs) and with different social forms, such as dyads, small groups and classroom groups. The scope of these tools' usage is very broad, but on the other hand setting up the scenario and having a pedagogically-based sequence of learning activities had to be done manually and without explicit process structure within the system. Thus the availability of formal descriptions of learning processes with IMS/LD and their interpretation within LD engines attracted our attention. The challenge is now to integrate a full-fledged learning support environment with a learning design engine without comprimising either of the two sides on the implementation level.

2.1 The basic idea

We propose an approach that aims at a clear separation of learning design engine, the specification and implementation of the learning flow (as LD documents) and learning

environments. In this proposal we assume that the learners interact excusively with the LSE without having to know anything about being "scripted" or "scaffolded" by the LDE resp. the LD document. According to Vogten, Koper, Martens, and Tattersall (2005) learning design engines can be considered as a collection of finite state machines that react to changes of properties with state transitions by sending events of a specific output alphabet. In the loosely-coupled connection of an engine with a learning support environment presented in figure 1, the engine controls the learning environment with output events (such as "start a new phase", event 1.), defined as a vocabulary for a set of environments, that are mapped by the environment to its existing functionality (such as "create new workspace", the configuration of the LSE through event 1.1). The learners interacting with the learning support environment create events (user action 2.), such as "phase is completed" (either directly or monitored by the LSE), that map to the input alphabet of the engine's state machines and are propagated to the LDE (message 2.1). The triggered state transition (message 2.2) causes the learning process to advance and will again trigger control messages (event 3.) to be accepted by the LSE. In that way we get the regulation cycle of figure 1 with the LDE and the LSE influencing each other's state. Using a generic vocabulary of communication primitives between the LDE and LSE has the advantage, that the LD document can be used with a variety of different LSEs without any changes to the document, given that the LSE can make use of primitives of the vocabulary.

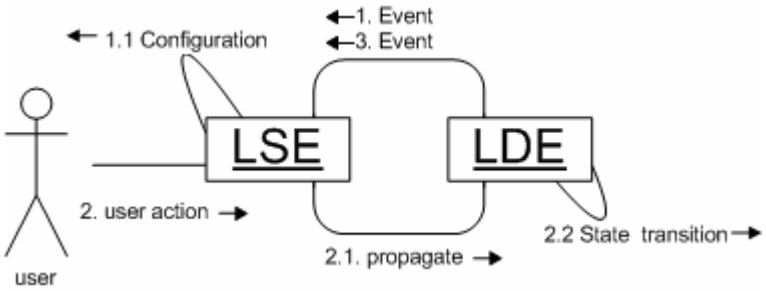

Figure 1: UML communication diagram for interaction schema between LDE and LSE

To explain the basic idea we introduce the simple learning flow sketched in Figure 2 as an example. First the students explore a phenomenon (e.g. the ballistic curve of a stone) and describe what they see. Then they model the observed phenomenon within a modelling environment. Every time a student states that the model is sufficient, it is frozen and each student has to vote if he approves the model. If there is a consensus the students present their model to the teacher. If they do not agree the modelling activity is continued.

This pattern is quite common for problem based learning. It can be varied, e.g. by changing the condition by which the students can finish the modelling phase. Alternatives to the

consensus decision are:

- a majority decision
- a teacher's decision
- a time constraint
- a semantic system constraint (e.g., the system checks the current model against a database of correct solutions and ends the modelling activity automatically if there is a match)

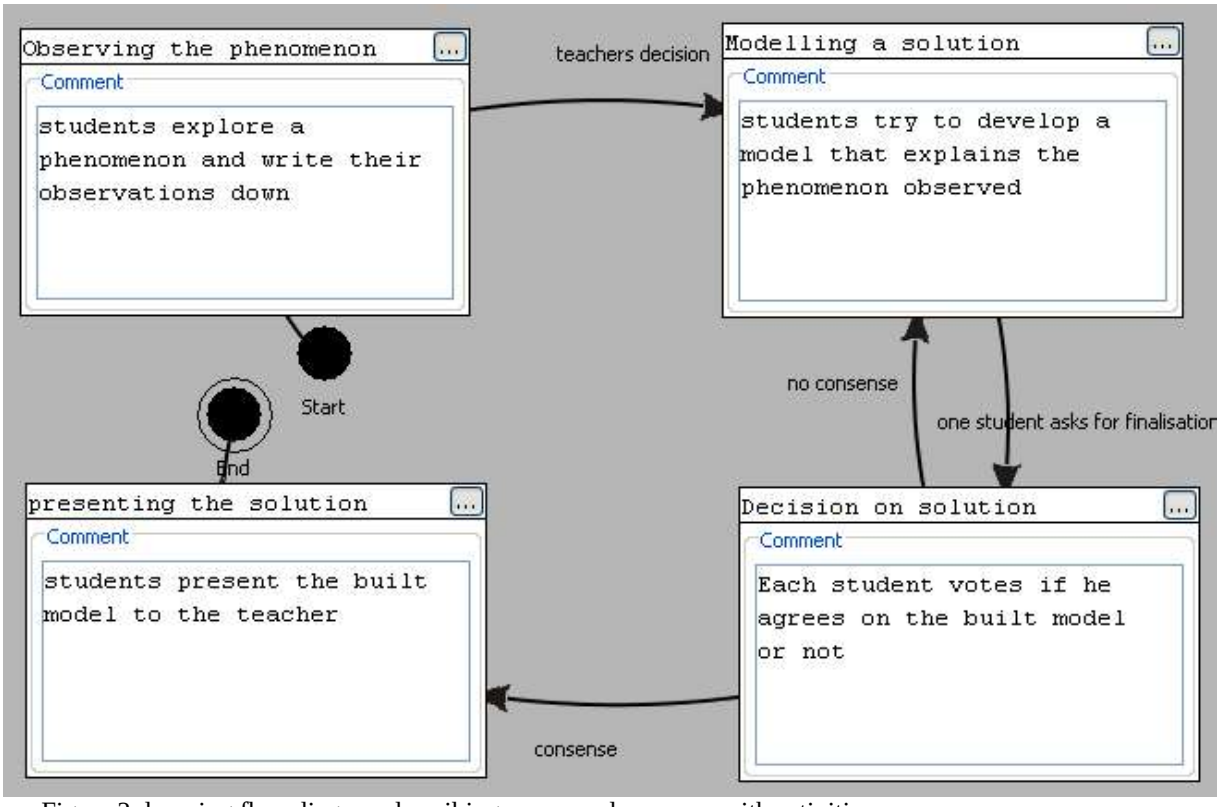

Figure 2: learning flow diagram describing an example process with activities supported by LSE(s)

To model this learning flow in IMS LD we will map each of the four learning activities to an activity element within IMS LD. In the beginning only the introductory activity is

shown, all other activities are hidden. The following properties are set up:

- current_activity ∈ {observing, modelling,presenting}
- voting_demanded ∈ boolean
- voting_active ∈ boolean
- voting_result ∈ real
- consensus_achieved ∈boolean

The properties, defined at IMS/LD level B, enable the dynamic regulation of the learning process within the LSE inititiated by the control messaged the LDE sends to it. The associated workspaces within our LSE are shown and hidden according to the orders the engine gives it. Depending on the state of the properties that are changed by the user's actions within the workspaces, e.g. conducting a voting, the engine's state machines are updated and lead potentially to state transitions. This is formalized in the Learning Design description as conditions, such as "if voting-result greater-than 0.8" or "if consensus-achieved is true". Evaluating the conditions causes new control events that are sent to the LSE to initiate a new setup there, e.g. going from state "Decision on solution" to state "Presenting the solution" when a consensus has been reached, which means that the presentation workspace will be made visible and properly configured with the produced solution.

2.2 The practical implementation

For the practical implementation we defined an architecture that brings together LSEs and LDEs without having to make substantial changes in either of the two components: the schematic overview of the architecture can be found in figure 3 and the components introduced have the following function:

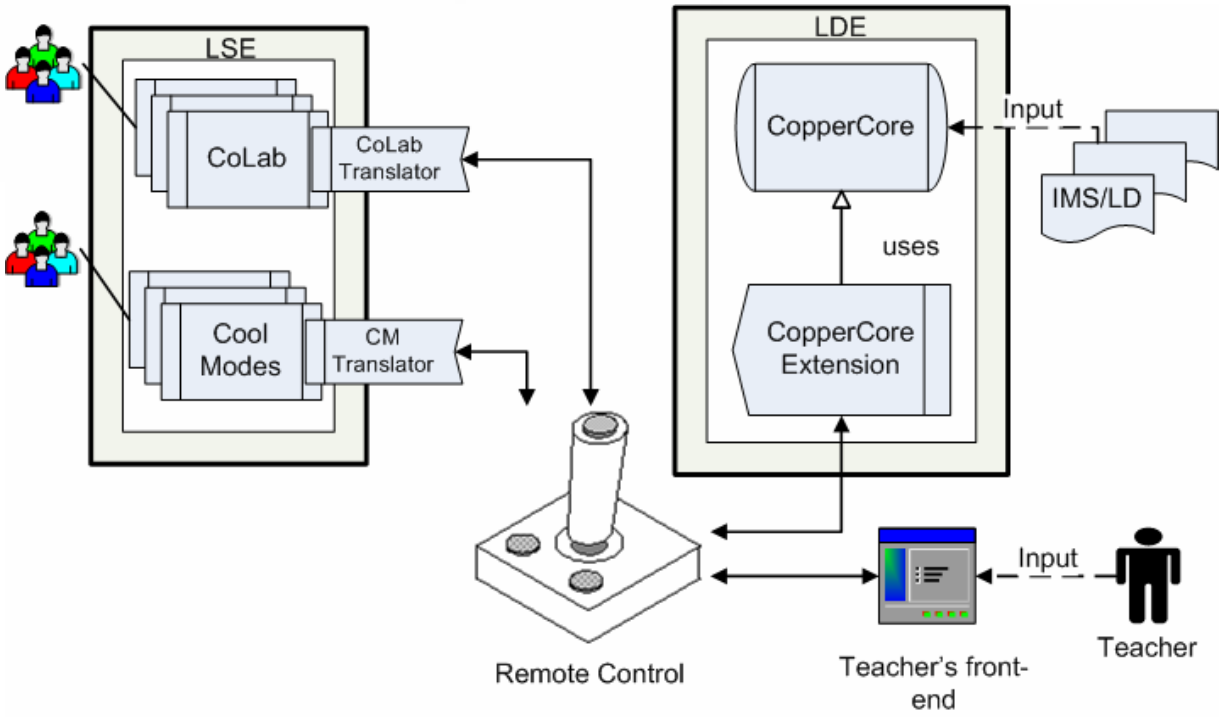

Figure 3: Remote Control Architecture for interaction between LDE and LSE

- Engine Extension (CopperCore Extension): this component extends the event propagation mechanism of the learning design engine, so that on state transitions within the engine, events are sent to the LSE to remotely control the learning process according to the LD document's description. This event is sent indirectly to the LSE via the Remote Control Component

- Remote Control Component: this component is the mediator between LDE and LSE; it maps events coming from the LDE to one or more communication primitives, that build the vocabulary for remotely controlling learning support environments, such as Co-Lab or Cool Modes. These "commands" are then sent to the "remote API" of the specific LSE.

- LSE Remote API (Translator): this interface accepts communication primitives that have been defined for a variety of different LSEs and maps these primitives to the specific functionality available in the concrete LSE. For example. the

communication primitive "ShowWorkspace for Decision Phase" could be mapped to calling the functionality "Make visible a workspace with title 'Decision on Solution' and add a Voting Plugin" in the Cool Modes environment (see figure 4). The primitive that has been sent out from the Remote Control Component to the subscribers of this primitive (all LSEs that understand the primitive) is then translated to a call of the respective functionality of the LSE; thus this can be considered a remote call of the LSE functionality by the Remote Control Component.

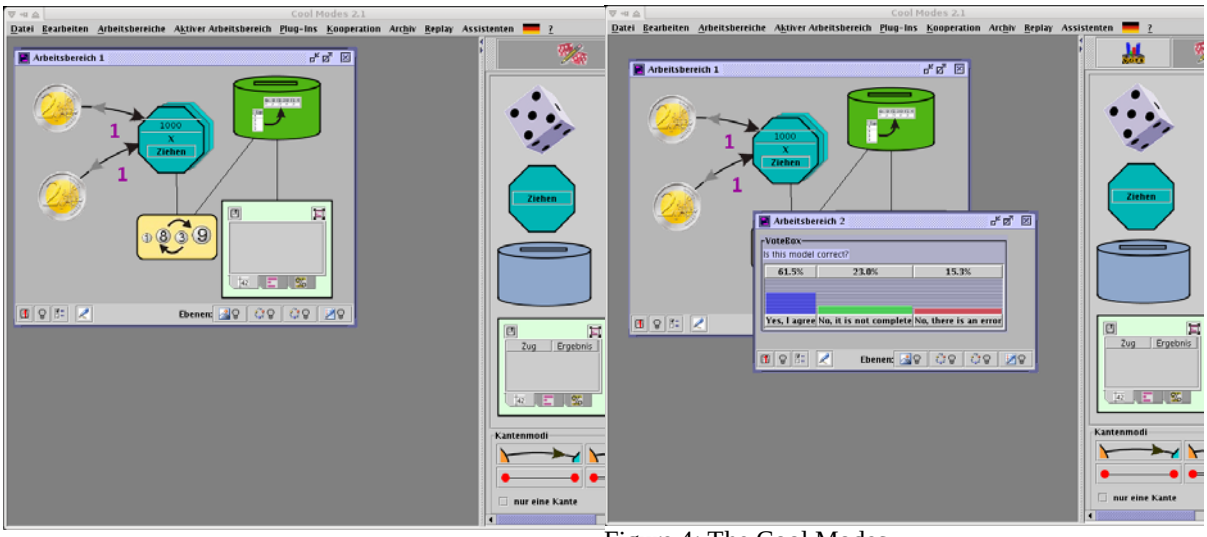

Figure 4: The Cool Modes learning environment before (left) and after (right) transmission of the communication primitive "ShowWorkspace for VotingPhase" from CopperCore Engine. In the right part the voting plugin was added (small icon in top right corner) and an additional window appeared to conduct the voting.

If some decisions in the learning process should be taken by the teacher at runtime, the remote control component is planned to be used in combination with a GUI frontend for the teacher to control the learning process, using the same functionality as the LDE. This analogous way of using the remote control is shown in figure 3 at the bottom right.

We implemented this architecture proposal prototypically using CopperCore as LDE and Cool Modes as LSE. Both creation of events in the engine extension and the mapping to Cool Modes functionality by a "Cool Modes translator" have been realized. The Remote Control Component encapsulates the mapping from LDE events to general communication primitives and uses the Java Message Service (JMS) to publish the primitives to the

translators subscribing to the Remote Control, i.e. the components that provide the remote interface to integrate Learning Support Environments into the learning design. At the moment we are in the process of generalizing and extending the vocabulary of communication primitives useful for different LSEs and the mapping to concrete LSEs, like Cool Modes and Co-Lab.

3 Conclusion and Outlook

The work presented here makes a first step in the integration of learning design descriptions and engines with complex learning support environments, that might be used for a variety of activities of a learning design. The basic idea is to keep the process execution and the practical activities separated, such that both engines and learning environments do not have to be modified substantially. This is achieved by definition of a vocabulary to let the LDE control remotely the LSEs that is used to regulate and configure the LSEs accoring to the state of the learning process. The central component to achieve this loose coupling is called "Remote Control Component".

In our presentation we assumed implicitly that the learners interact exclusively with the LSE without knowing anything about the process state of the learning design. When, in contrast, the learners should be made aware of the process, e.g. to give them a help in their orientation of what they have already achieved, a tool that visualizes the current state of the engine (much like the conventional function of a player) could be introduced. This process awareness tool could also have control elements for the process (such as "proceed to the next phase") to initiate events and thus state changes within the engine. Similarly a teacher could use the Remote Control Component (see figure 3 lower right) via a teacher frontend in case the teacher has to decide personally on the regulation of the learning process, e.g. if some details cannot be described in advance in the LD document.

Acknowledgements: We thank our student Benedikt Roth for his implementation work on integrating CopperCore and Cool Modes. Additionally we thank all the people that helped us to refine the idea of the "remote control" in discussions, such as our partners in the COSSICLE European Research Team in the Kaleidoscope Network of Excellence.

4 References

Co-Lab (2005): Collaborative Laboratories, European Founded IST-Project No. IST-2000-25035, Accessed online on 3 May 2005 at: www.co-lab.nl,

CopperCore (2005), The IMS Learning Design Engine, Accessed online on 3 May 2005 at: coppercore.org/

Hoppe, H. U. & Gaßner, K. (2002). Integrating collaborative concept mapping tools with group memory and retrieval functions. In G. Stahl (Eds.), *Computer Support for Collaborative Learning: Foundations for a CSCL Community* . Boulder, USA: distrib. Lawrence Erlbaum.

Pinkwart, N. (2003). A Plug-In Architecture for Graph Based Collaborative Modeling Systems. In U. Hoppe, F. Verdejo & J. Kay (eds.): *Proc. of Artificial Intelligence in Education*, Amsterdam, IOS Press.

Suthers, D., Weiner, A., Connelly, J. & Paolucci, M. (1995). Belvedere: Engaging students in critical discussion of science and public policy issues. In Greer, J. (Ed.), *Proc. of AI-ED 1995*. Washington DC (USA).

Vogten, H., Koper, R., Martens, H., and Tattersall, C. (2005). An Architecture for Learning Design Engines. In Koper, R., Tattersall, C. (eds.): *Learning Design*. Springer: Berlin.

WISE (2005) : the web-based inquiry science environment, Supported by the National Science Foundation NSF, Accessed online on 3 May 2005 at: wise.berkeley.edu