A Model-driven Approach to the development of a PBL Script Editor

Yongwu Miao[1*], Mohammed Samaka[2], John Impagliazzo[3], Ying Wang[1], Disi Wang[1], Ulrich Hoppe[1],
[1]*Department of Computational and Cognitive Sciences, University of Duisburg-Essen, Germany*
[2]Computer Science and Engineering Department, Qatar University, Doha, Qatar
[3]Department of Computer Science, Hofstra University, New York, U.S.A.
[*]miao@collide.info

Abstract: *Designing a pedagogically sound and technically executable collaboration script is a highly complex, time-consuming and error-prone task. This paper presents a model-driven approach to enable practitioners to design online PBL courses. Adopting this technical approach, we developed a PBL scripting language that provides natural concepts for the teacher to use in PBL practices. Based on the PBL scripting language, we developed a PBL script editor that facilitates teachers to design, communicate, customize, and reuse PBL scripts. In addition, it provides functions to transform a PBL script to a unit of learning (UoL) represented in IMS Learning Design (LD), which can be executed in an IMS LD run-time environment to scaffold PBL processes.*

Keywords: Model-driven Approach, CSCL, collaboration script, PBL script, IMS LD, Unit of Learning

1. Introduction

Boud and Feletti (1991) described Problem-Based Learning (PBL) as an approach to structuring the curriculum which involves confronting students with problems from practice which provide a stimulus for learning. In a PBL process, students usually work in teams to analyze the problem in a manner that allows their ability to reason and apply knowledge to be challenged, evaluated and developed. Bringing real-life context and technologies to the curriculum through a PBL approach encourages students to become independent workers, critical thinkers, problem solvers, lifelong learners, and team workers (Barrows 1985, Hmelo & Eberbach 2012).

Despite the potential for success, there are a number of impediments to PBL's diffusion, because implementing effective collaborative PBL is a challenging task. As research on collaborative learning has repeatedly shown, learners typically do not engage in these "high-level" collaboration processes without guidance (e.g., Weinberger, Stegmann, Fischer, & Mandl, 2007). Thus, a crucial question for research is how collaborative learning can be supported in order to stimulate such high-level collaboration processes and learning outcomes. The notion of "collaboration script" emerged from the observation that there is a need to design teamwork in such a way that productive interactions emerge such as identifying problems, argumentation and assessment. This expression has been used to designate scaffolds structuring face-to-face collaboration (O'Donnell and Dansereau 1992) and computer supported collaboration (Dillenbourg 2002). Computer supported collaborative learning (CSCL) scripts are considered an effective means of facilitating specific interaction patterns in CSCL situations (see Fischer, Kollar, Mandl, & Haake, 2007). The motivation of this research is applying collaboration script in PBL and using PBL scripts to structure and support technology-enhanced, problem-oriented, collaborative learning processes.

Although a lot of researches on collaboration script have been reported in literature (e.g., Dillenbourg, 2002; Miao, et al. 2005; Dillenbourg & Tchounikine, 2007; Harrer, et. Al. 2009), using PBL scripts involves some challenging issues. First of all, a PBL scripting language is needed to specify a PBL process. Although some scripting languages such as LDVS (Agostinho, et al. 2008) can help teachers to understand and communicate inspirational PBL scripts no PBL scripting language is available. Such a scripting language as LDVS usually has no explicit syntax and semantics specified. The pedagogy and the rationale of the actual design are described informally through the use of textual description and visual diagrams. Another issue is whether a PBL script can be used to facilitate specific interaction patterns and scaffold students in a CSCL environment. Some scripting languages such as IMS Learning Design (LD 2003) and Learning Design Language (LDL) (Martel and Vignollet 2008) have explicit syntax and semantics. A PBL process can be specified by using these scripting languages as a formal script. Such an executable PBL script can be processed automatically by a machine to guide students to work collaborative following the prescribed collaborative

patterns and to provide specified resources and services for an effective learning. Finally, whether an easy to use scripting tool is available to facilitate the development of executable PBL scripts? For example, an IMS LD compatible authoring tool or a LDL authoring tool provides insufficient means for ordinary teachers to develop and understand a PBL script, because these scripting languages are pedagogy-neutral and the language compatible scripting tools lack components required for modeling PBL scripts because they were not specifically designed to support PBL.

To address the challenges mentioned above, we propose a model-driven approach (MDA) for supporting the development of executable PBL script. Though developing a prototype, we argue that this approach and the associated tool are technically feasible.

2. An Overview of Our Model-driven Approach

Researchers, developers and end-users with specific domain expertise are more and more requiring computational processes to allow them to complete some task. MDA is such an approach that provides higher levels of abstraction to allow such users to focus on the problem, rather than the specific solution or manner of realizing that solution through lower level technology platforms (Schmidt, 2006). Domain Specific Modeling language (DSML) is often used to realize MDA. It allows domain users to think in terms closer to the problem domain when specifying their systems, by providing a way to think at the same abstraction level of the problem under consideration. The gap between the real problem and the mental model is reduced with respect to the generic approach of using General Purpose Languages such as the Unified Modeling Language (UML).

By applying the MDA paradigm, we developed a PBL scripting language and a PBL script editor. The overview approach is illustrated in Figure 1. The vocabularies and syntax of the PBL scripting language are defined through analyzing the commonalities and differences of various PBL theoretical models and various PBL practices from the perspectives of process modeling. Rather than taking abstract and/or technical concepts such as “activity” and “property”, PBL-specific concepts such as “identifying” and “learning issue” are selected so that teachers can understand and use them because the PBL scripting language is similar to one they daily used to describe a PBL process.

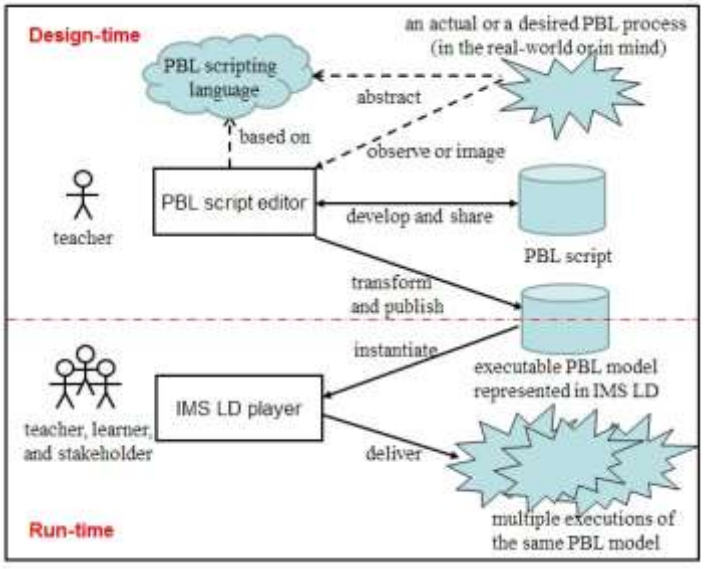

Figure 1: A Model-driven Approach

In order to facilitate PBL practitioners to represent a PBL design easily, we develop a graphical PBL script editor in the light of the PBL scripting language. It provides structure and guidance for PBL practitioners to create, communicate, customize, and reuse a PBL script, a description of a PBL process represented in the underlying PBL scripting language. The editor also provides functions to transform a PBL script into a unit of learning (UoL) represented in IMS LD. Although IMS LD is a DSML as well, its abstract level is just a little higher than UML because the main concepts (e.g., learning-activity and local-personal-property) are still generally and abstractly defined (see Koper, & Olivier, 2004).

However, a UoL is executable and the transformed PBL script can be played by practitioners to scaffold PBL practices in an IMS LD player such as SLeD (Weller et.al, 2006) and Astro (aster homepage).

3. Development of a PBL Scripting Language

As mentioned above, the PBL scripting language is a DSML designed for the PBL practitioner to represent a PBL process. We developed the PBL scripting language by abstract well-known PBL models such as Barrows model (Barrows, 1989), Woods' model (Woods, 1996), and Seven Jumps (Wilkerson, 1996) and many best PBL practices. Using the language, a PBL process can be represented as a sequence of phases (e.g., *problem-engagement, problem-analysis, aim-and-plan, research, problem-resolution,* and *evaluation*). In a phase, relevant activities (e.g., *presenting, identifying, planning, investigating, generating, synthesizing, debriefing, reflecting-on,* and *commenting-on*) will be performed *individually*, *collaboratively* or *in-turn* for achieving the goal of the phase. Because we focus on supporting PBL practices in a typical class, in which all students may be divided into several small groups. The role or actor(s) of an activity could be assigned to *all-students-in-the-class, all-groups-in-the-class, all-students-in-each-group, all-students-in-a-specific-group, a-specific-student,* or *a teacher*. Associated artifacts (e.g., *problem-statement, learning issues, learning task, information, solution,* and *assessment*) are produced and used in activities. Some relations among these concepts are specified as constraints. For example, in the *problem-engagement* phase only some types of activities (e.g., *presenting, brainstorming, discussing, identifying* and *formulating*) will be performed and some kinds of artifacts such as *case, scenario, phenomenon,* and *problem-statement* will be dealt with. A problem-statement can be stipulated as an outcome of the *problem-engagement* phase and can be transferred into *problem-analysis* phase and *problem-resolution* phase. Moreover, activity sequence and artifact transferring/distribution can be specified.

Additional elements and attributes are defined in the PBL scripting language to enhance the expressiveness of the language. For example, generic attributes such as *title, objectives, description* and *completion-condition* are defined for modeling an activity. Some activities such as *debating* and *discussing* contain specific elements and attributes. For example, *communication-tool* is an element and *communication-mode* is an attribute. A *communication-tool* could be a *chat, a whiteboard, a wiki,* or *a discussion-forum*. The value of the attribute *communication-mode* could be *synchronous* or *asynchronous*.

4. Development of a PBL Scripting Tool

In order to facilitate the practitioner to design PBL courses with the PBL scripting language, we developed a PBL Script Editor based on SCY-SE (Wichmann, Engler, & Hoppe, 2010).

As illustrated in Figure 2, the editor consists of four parts. The menu bar on the top lists basic functions and the edit bar on the bottom provides functions to edit text, scribe, and image. The central area contains edit-space on the left and tool-space on the right. The user can use organizational role definition tool in the tool-space to specify how many students there are in a class and to divide them into small groups (or not). An important tool is the palette. When creating a PBL script the user can define phases and artifacts in the process-page of the edit space by dragging the phase-node and the artifact-node from the process-palette and dropping on the process-page. Phase-sequence and artifact-transferring/distribution among phases can be specified by clicking the draw-edge button or delete-edge button and then connecting two related nodes on the process-page. When the user double-clicks a created phase-node on the process-page, the corresponding phase-page will open and the phase-palette will open as well. The user can specify the internal process structure within the phase by dragging and dropping role-node, activity-node, and artifact-node and by manipulating edges. When specifying a phase, a role, an activity, or an artifact, the user can select a type from a list of recommended types. For example, in the phase *problem-resolution*, the user can choose one of recommended activity type such as *brainstorming, clarifying, commenting-on, comparing, debating, debriefing, discussing,* and *generating*, which are usually performed in this phase. Additional information can be specified through opening the detail definition windows. For example, the user can click right-mouse-button over an activity-node and choose a menu item to specify

general information such as *title*, *objectives*, and *description*, to input specific materials for this activity, or to choose an activity-completion-condition.

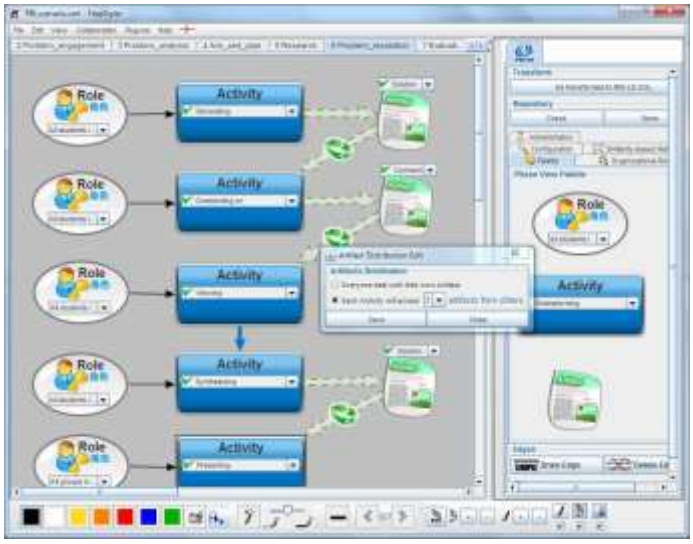

Figure 2: The user interface of PBL Script Editor

The diagram shown in Figure 2 represents a design of a *problem-resolution* phase, in which each row has two or three nodes forming a sentence. For example, the types of the role-node, activity-node and artifact-node in the first sentence are specified as *all-students-in-the-class*, *generating*, and *solution*, respectively. The generating activity can be specified in details by providing further information for *title* and *description* and by choosing *individually* as the work-mode and *half-hour* as the completion-condition. The types of three nodes in the second sentence are specified as *all-students-in-each-group*, *commenting-on*, and *comment*, respectively. Note that the dash-line arrow connecting the *solution* node of the first sentence and *commenting-on* node of the second sentence represents an artifact-distribution. By clicking the right-mouse-button over the arrow, the user can choose a menu-item and open a dialog (see it in the middle of Figure 2) to define an artifact-distribution-policy. In this situation, the user defines that each *commenting-on* activity has three *solution* artifacts as input. Similarly the *comment* artifacts are transferred as the input of the viewing activity in the third sentence. The *artifact-distribution-policy* is specified as sending *comment* artifact back to the owners of the *solution* artifact. Then, the role *all-students-in-each-group* is assigned to a *synthesizing* activity, which work-mode is defined as *collaboratively* and which output is a *solution* artifact. The final sentence is specified as the role *all-groups-in-the-class* performs a *presenting* activity *in-turn*. Obviously, this segment of process contains various process structures and complex control-flow and data-flow. The PBL scripting language and the PBL scripting editor enable to represent complex process structure and process controls implicitly.

By adopting MDA, we define the semantics of the PBL scripting language through translating semantics, where the abstract syntax of the PBL scripting language is mapped into the abstract syntax of the IMS LD. Usually we can map the constructs of the PBL scripting language to the constructs of the IMS LD and maintain their basic relations, because most concepts used in the PBL scripting language are specialized concepts of IMS LD. For example, the constructs of the PBL scripting language such as *PBL-process*, *problem-engagement*, *all-students-in-the-class*, *formulating*, *problem-statement*, *chat*, and *case* are just specialized constructs of IMS LD such as *play*, *act*, *role*, *learning-activity*, *property*, *service*, and *learning-object*, correspondingly. However, sometimes it is challenge to translate concepts (e.g., an artifact distribution policy) properly, because there is no a simple mapping to a construct of the IMS LD. In these cases we carefully designed mapping rules to translate the concepts into a combination of constructs of the IMS LD with certain constraints. Because of the limited space of the paper we cannot present more technical details and more features of the PBL script editor.

5. Discussion

This section discusses two issues concerning the model-driven approach and the PBL script editor. First, few researches were reported recently on applying MDA and DSML to facilitate the design of e-learning platforms. For

example, Laforcade (2010) and Dehbi et al. (2013) used MDA to automatic implementation of new learning management systems. It is worth to note that the collaborative problem-based meta-model (Thierry, et al. 2008) raised the level of abstraction from programming to a level similar to IMS-LD. It has no PBL-specific constructs as those defined in our PBL scripting language. Our PBL scripting language raised the level of abstraction from IMS LD to PBL application domain and then transformed a PBL script into an executable IMS LD model. Second, normal graphical scripting languages and associated tools (e.g., Harrer, et. al. 2007; Neumann and Oberhuemer, 2009) enable the development of PBL scripts. However, it is difficult, if not impossible, for practitioners to understand and use this modeling language, because it requires sound pedagogical knowledge as PBL and overhead technical knowledge as process modeling (Neumann and Oberhuemer, 2009). Our language and tool provides PBL-specific concepts and means for PBL practitioners to use intuitively.

6. Conclusions and Future Work

In order to facilitate PBL practitioners to structure and conduct PBL activities easily and to gain benefits from exploiting open e-learning technical specification, we adopted a model-driven approach to support the development of collaboration script. In this paper, we outline our approach to apply MDA paradigm to the development of a PBL scripting language and an associated graphical PBL script editor. Through developing the scripting language and the editor, we demonstrated that this approach is technical feasible. The users can use the editor to describe a PBL script even though they don't know how to handle complex process controls such as synchronization and loop and cannot explicitly represent complex data-flow and various forms of activities and collaboration shifting among the levels of individual, small group, and class. The transformation functions can handle the technical complexities and make mappings from the high-level concepts used in the PBL scripting language to the low-level concepts of IMS LD. In particular, the user-friendly user interface and design guidance (e.g., the hints for choosing a type of activity or artifact) are very useful for understanding and developing a PBL script.

So far, we have focused mainly on designing and developing the tool to demonstrate the technical feasibility. According to our project plan, we will improve the PBL scripting language and the PBL script editor on the aspects of completeness, flexibility, usability, and reusability. For example, we will add more PBL-specific design guidance, design options, services and templates in the editor to improve the usability. We plan to research the methods and algorithms to measure the similarity of the PBL scripts and to support similarity-based retrieval. Finally, we will conduct evaluation to empirically assess the added value of the approach and the PBL scripting tool in the PBL practice.

Acknowledgements

This work has been supported under the grant ID NPRP 5 - 051 - 1 – 015 for the project entitled PLATE, which is funded by the Qatar National Research Fund (QNRF).

References

Agostinho, S., Harper, B., Oliver, R., Hedberg, J., & Wills, S. (2008). A visual learning design representation to facilitate dissemination and reuse of innovative pedagogical strategies in university teaching. In L. Botturi & T. Stubbs (Eds.), *Handbook of visual languages for instructional design: Theories and practices*. Information Science Reference.

Astro Player, at http://tencompetence-project.bolton.ac.uk/ldruntime/index.html

Barrows, H. (1989). *The Tutorial Process*. Springfield Illinois: Southern Illinois School of Medicine.

Boud, D., and Feletti, G. (1991). *The Challenge of Problem-Based Learning*, London: Kogan Page.

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, The Netherlands: Open Universiteit Nederland.

Dillenbourg, P., & Tchounikine, P. (2007). Flexibility in macro-scripts for computer-supported collaborative learning. *Journal of Computer Assisted Learning, 23,* 1–13.

Dehbi, R., Talea, M., and Tragha, A. (2013) A Model Driven Methodology Approach for e-Learning Platform Development, *Inter. Journal of Information and Education Technology*, *3*(1): 10-15.

Fischer, F., Kollar I., Mandl, H., & Haake, H. M. (2007). *Scripting computer-supported collaborative learning: Cognitive, computational and educational perspectives*. New York: Springer.

Harrer, A., Malzahn, N. and Hoppe, U. (2007) "Graphical Modeling and Simulation of Learning Designs, Supporting Learning Flow through Integrative Technologies," in T. Hirashima and U. Hoppe and S. Young eds. *Frontiers in Artificial Intelligence and Applications*, Vol. 162. IOS Press, Amsterdam.

Hmelo-Silver, C.E., Derry, S.J., Bitterman, A. and Hatrak, N. (2009).Targeting Transfer in a STELLAR PBL Course for Preservice Teachers, *The Inter. Journal of Problem-based Learning*, 3(2), 24–42.

Hmelo-Silver, C.E. & Eberbach, C. (2012). Learning theories and problem-based learning. In S. Bridges, C. McGrath, & T. Whitehill Eds. *Researching problem-based learning in clinical education: The next generation*, pp. 3-17, New York: Springer.

IMS Learning Design Specification, at http://www.imsglobal.org.

Kolodner, J. L., Crismond, D., Gray, J., Holbrook, J., Puntambekar, S. (1998). Learning by Design from Theory to Practice. *Proc. ICLS 1998*, pp. 16-22. Atlanta, Georgia.

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

Laforcade, P. (2010). A Domain-Specific Modeling approach for supporting the specification of Visual Instructional Design Languages and the building of dedicated editors. *Journal of Visual Languages & Computing*. Volume 21, Issue 6, Pages 347–358.

Martel, C., and L. Vignollet. (2008). Using the learning design language to model activities supported by services. *International Journal of Learning Technology,* 3(4): 368–87.

Miao, Y., Hoeksema, K., Harrer, A. and Hoppe, U., (2005). CSCL Scripts: Modeling Features and Potential Use. *Proceedings of Computer Supported Collaborative Learning (CSCL'05) conference*, pp. 423-432, May 30 – June 4, 2005, Taipei, Taiwan.

Neumann, S. and Oberhuemer, P. (2009). User Evaluation of a Graphical Modeling Tool for IMS Learning Design. *Proc. ICWL 2009*, pp. 287-296. Germany: LNCS 5686, Springer.

O'Donnell, A. N., & 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.), *Interactions in cooperative groups. The theoretical anatomy of group learning* (pp. 120–141). Cambridge, MA: University Press.

Schmidt, D. C. (2006). Introduction: Model-Driven Engineering. *IEEE Computer*, 39(2), 25-31.

Scardamalia, M., Bereiter, R. S., Swallow, M. J., and Woodruff (1989). Computer-supported intentional learning environment. *Journal of Educational Computing Research*, 5, 51-68.

Suebnukarn, S. and Haddawy, P. (2006). A Bayesian Approach to Generating Tutorial Hints in a Collaborative Medical Problem-Based Learning System. In *Artificial Intelligence in Medicine*.

Thierry, N., Pierre-Andr é, C., Xavier, L.P., and Pierre, L. (2008). Applying Model Driven Engineering Techniques and Tools to the Planets Game Learning Scenario, *JIME Special Issue: Comparing Educational Modelling Languages on the "Planet Game" Case Study*, available at: http://jime.open.ac.uk/article/2008-23/367

Weller, M., Little, A., McAndrew, P., & Woods, W. (2006). Learning Design, generic service descriptions and universal acid. *Educational Technology & Society*, 9 (1), 138-145.

Weinberger, A., Stegmann, K., Fischer, F.,&Mandl, H. (2007). Scripting argumentative knowledge construction in computer-supported learning environments. In F. Fischer, H.Mandl, J. Haake,& I. Kollar (Eds.), *Scripting*

computer-supported communication of knowledge – cognitive, computational and educational perspectives (pp. 191–211). New York: Springer.

Wichmann, A., Engler J., & Hoppe H. U. (2010). Sharing educational scenario designs in practitioner communities. *Proc, of the 9th International Conference of the Learning Sciences 2010*, 750-757.

Wilkerson, L. & Gijselaers, W.H. (Eds.) (1996). *Bringing Problem-Based Learning to Higher Education: Theory and Practice*. San Francisco: Jossey-Bass Publishers.

Woods, D.R. (1996). *Problem Based Learning: How to Get the Most from PBL*, McMaster University.