A Flexible System and Data Model for the Representation and Management of PBL Scripts

Disi Wang[a*], Yongwu Miao[a], Ulrich Hoppe[a] & Mohammed Samaka[b]
[a]*Department of Computational and Cognitive Sciences, University of Duisburg-Essen, Germany*
[b]*Computer Science and Engineering Department, Qatar University, Doha, Qatar*
*wang@collide.info

Abstract: As problem-based learning (PBL) is becoming more and more popular, there is also a growing interest in developing and using technologies in the implementation of PBL. However, teachers may have difficulties to design a pedagogically high-quality and technically executable, online or blended PBL process on their own, because they lack appropriate expertise in learning theories and design methods as well as a deeper understanding of the potential affordances of technologies. From this premise, we are committed to developing and testing methods and tools to support the design and delivery of online or hybrid PBL processes with high flexibility and a low threshold of usage requirements. As a first milestone towards this goal we have developed a web-based PBL authoring tool for facilitating teachers to design, represent, understand, communicate, customize and reuse online or blended PBL processes. This paper presents a technical approach to the PBL authoring tool and focuses on supporting the representation and management of PBL scripts, computational descriptions of PBL process. In comparison with other tools and technical approaches, it is concluded that a combined use of semi-structured data management and a unified data format appears to be a promising approach to effectively and efficiently support the representation and management of PBL scripts.

Keywords: PBL, learning design, PBL authoring tool, PBL script, semi-structured data, data management, graphical representation, web-based application

1. Introduction

Problem Based Learning (PBL) is a learning method to structure the curriculum in such a way as to involve confronting students with problems from practice as a stimulus for learning (Boud & Feletti, 1998). It engages learners in an active, collaborative, student-centered learning process that develops critical thinking, problem-solving, team-work, and self-directed learning abilities needed to meet the challenges of life and career in our increasingly complex environment (Hmelo-Silver & Eberbach, 2012). PBL has been successfully used in various different domains and its benefits have been largely demonstrated (Savery, 2006). The application of new technologies such as Web 2.0 and virtual collaboration environments can enrich and improve implementations of PBL (Kaldoudi et al., 2008). Nevertheless, to achieve its full power, an online or blended PBL process needs to be well designed, and a sound online or blended PBL process may be a collaborative product of years of research, application, assessment and redesign. Usually, designing such a PBL process is an intensively mental work and an implicit process. Traditionally, the teacher individually constructs a PBL design based on personal experience and represents it in natural language as a course plan or lesson plan on paper. Teachers may have neither appropriate expertise in learning theories and design methods nor a deep understanding of the potential affordances of technologies. They lack guidance to design a high-quality and technology-supported PBL plan in a specific context in order to benefit from PBL and new technologies.

Our general goal is to study and develop methods and tools to support the design and delivery of online or hybrid PBL processes in an effective, efficient and flexible manner. As a milestone to achieve this goal we developed a web-based PBL authoring tool that aims at facilitating teachers in design, representation, understanding, communication, customization and reuse of online or blended PBL processes. We try to help teachers to make the implicit design process explicit in order to improve design quality and to represent a traditionally informal description as a formal model for scaffolding and orchestrating a PBL process by the computer. This paper presents our technical approach to develop a web-based PBL authoring tool. We claim that semi-structured data management and a unified semi-structured data form can make up an effective combination to technically support the representation and management of PBL design.

The rest of the paper is organized as follows: Section 2 describes a typical PBL process script represented in natural language and characterizes a PBL design. Section 3 presents the implementation and the functionalities of the PBL authoring tool to give an impression of our tool. Section 4 identifies technical requirements that how to represent and manage PBL scripts in order to implement the tool. Section 5 compares our tool with related work on the aspects of user interface and data management. The final section summarizes our work and indicates the future work.

2. Requirements to Represent and Manage PBL Scripts

PBL can be conducted in a number of ways. There are different PBL models such as McMasters PBL model (Woods, 1996), the Maastricht “seven jump” model (Barrows, 1996), the Aalborg “Problem Oriented Project Pedagogy (POPP)” model (Dirckinck-Holmfeld, 2002), Mills “five-stage” model (Mills, 2006), and SALFORD model (McLoughlin & Davrill, 2007). According to concrete situations such as the learners’ prior domain knowledge and PBL skills, the topics to be learned and the problem used to drive learning, and the availability of learning technologies, a PBL process can be designed to fit the specific learning context. In order to illustrate how a PBL process is usually described informally in natural language, we take a segment of “seven-jump” model as an example through this paper from (Maurer and Neuhold, 2012):

> “To get students started on a certain topic, they are confronted with an assignment that … outlining the problem or asking for a specific task to complete. … Students are supposed to have read and looked at this assignment already before their tutorial (or during the break), so that they can start with **clarifying terms and concepts**. This first step guides students mentally into the topic, and by discussing unknown words or concepts it is ensured that all students understand the text as it stands and that the group shares ideas about illustrations that might be part of the assignment. This first step provides a common starting point and leads the group into the topic. In the next step, the whole group agrees on the **formulation of the problem statement** that frames the whole assignment, provides a title for the session, and makes the group agree on what the general impetus of the assignment is about. Problem statements can take the form of more traditional titles, but are sometimes also formulated as broader research questions or provoking statements. The problem statement should trigger the next step of the **brainstorm**. … Everything is allowed during this step, and ideas are collected unquestioned at the whiteboard (i.e. there are no wrong ideas; everyone should be allowed to follow her/his own ideas). … The outcome of the brainstorm is noted on the whiteboard by the secretary that during the next (fourth) step should be **categorized and structured** by the students. … but by structuring the brainstorm students categorize keywords that fit together and in this way they find common patterns that in the next step will allow for the formulation of specific questions. As last step of the pre-discussion, students agree on the **formulation of common learning objectives**, by referring to the brainstorm and the now structured collection of ideas that they have noted on the whiteboard. …”

This PBL model consists of several steps (marked in bold, here only shows five steps for the reason of space) which includes *clarifying terms and concepts*, *formulation of the problem statement*, *idea brainstorming*, *categorizing and structuring ideas*, and *formulation of learning objectives*, etc. In each step one or several activities are performed by the facilitator, group, individual learner or stakeholders, e.g. scientific staff. For instance, according to the quoted test, after a problem statement was formulated in step 2, the third step *idea brainstorming* will be triggered. Assume that this learning model will be performed by a class which is divided into 3 groups and instructed by one facilitator, then in this step, a specific activity sequence could be 1) based on the problem statement formulated in the previous step, each group creates a basic hypothesis; 2) from this basic hypothesis each group proposes their tentative solution; 3) the facilitator views these solutions and give his or her feedback. Each group will improve their tentative solutions based on the feedback. In this step, there are some artifacts will be produced: three groups of basic hypotheses, three groups of tentative solutions and the facilitator’s feedback. All this work could be carried out by using an (online) whiteboard. Figure 1 illustrates the process above.

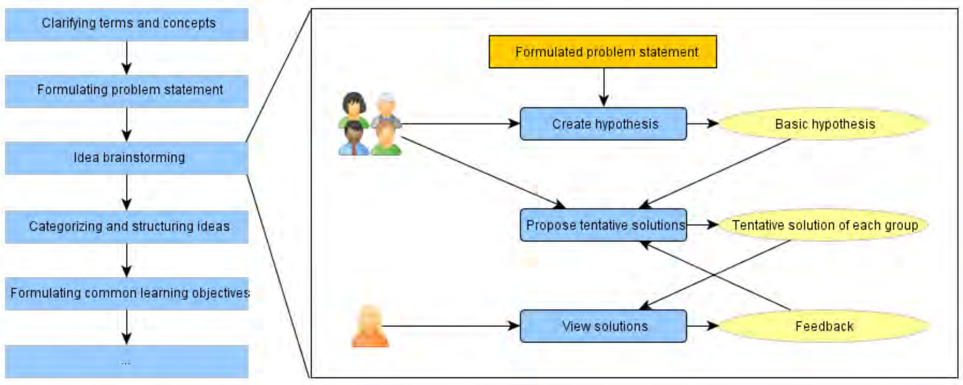

Clarifying terms and concepts
Formulating problem statement
Idea brainstorming
Categorizing and structuring ideas
Formulating common learning objectives
...
Formulated problem statement
Create hypothesis
Basic hypothesis
Propose tentative solutions
Tentative solution of each group
View solutions
Feedback

Figure 1. Structure demonstration of a PBL process.

As it is shown in the figure 1, there are several top level steps in the left, which is also called phases. The phases have a process sequence from top to bottom. Then in each phase, *idea brainstorming* for instance, there is a second level of description, or phase internal definition. Teacher needs to organize actors first and then assign actors to different activities. Some activities will produce artifacts or need learning resources. The activities are also under a process sequence.

Existing ICT applications for PBL still lack flexibility and usability/understandability by teachers who are not PBL experts. Either some applications are easy to use because they have well-designed PBL process pattern inside, e.g. STELLAR (Hmelo-Silver et al., 2009) and ePBL (Ali & Samaka, 2013), but too rigid to apply different PBL models; or some applications are flexible enough, but difficult for teachers who lack a deep understanding of PBL processes. For example, teachers need to figure out which activities are appropriate for a certain phase; what kinds of artifacts should be there as temporary outcome or final outcome for certain phase and so on. Since process scripts are usually just embedded in teacher's practice and they tend to be implicit. Thus in educational practice, PBL processes mostly are executed only based on a social protocol and a manual configuration and management of various application tools and digital or non-digital artifacts. As a consequence, there is a need to support teachers in representing PBL processes more explicitly and formally.

In order to overcome these barriers and to enable computerized support of different models of online PBL, one possible way is to adopt the concept of CSCL script. As we know, this concept has been used to structure computer supported collaboration (Dillenbourg, 2002). It has been considered an effective means of facilitating specific interaction patterns in CSCL situations (Fischer et al., 2007). Numerous approaches to representing CSCL scripts and CSCL scripting languages have been reported in the literature (e.g. Dillenbourg, 2002; Miao et al., 2005; Miao et al., 2007; Dillenbourg & Tchounikine, 2007; Harrer et. al. 2007). Our approach is particularly inspired by Miao et. al. (2000), and we proposed our PBL specific scripting language, namely PBL scripting language. Rather than using generic vocabularies, our scripting language only relies concepts that teachers use daily to describe PBL processes such as *problem engagement*, *identify learning issue*, *generate solutions*, *evaluate acquired knowledge*, etc. Also, as a kind of Domain Specific Modeling Language (DSML), these vocabularies and rules are specified by PBL domain experts taking into account different PBL models. In this way, teachers can be enabled to make use of different PBL models to create different but correct and computer understandable PBL scripts. As a result, teachers' mental scripts become representable, manageable and runnable online or as blended PBL course plans.

Therefore, there is a requirement to find a way of providing a web-based application which is aimed at empowering teachers to represent, improve, understand, share, reuse and manage online or blended PBL scripts in a design-time basing on the applying of our PBL scripting language.

3. A Web-based PBL Authoring Tool

In order to fulfill the requirement mentioned above, we have developed a web-based PBL authoring tool whose basic concept has already been introduced and evaluated. Miao et al. (2013) provide evidence that this tool is understandable and usable by teachers to develop online PBL course plans. Taking up the example in section 2, figure 2 shows the three major User Interface elements of this tool. They separately represent the organization design UI, the script level design UI and the phase level design UI. The most important functional area is the middle area which contains, from left to right, 1) script file management panel (for scripts management, sharing, reusing, etc.), 2) meta-type pool (which provides corresponding meta-types for different types of edit-spaces according to the PBL scripting language meta-model), 3) graphical script edit-space and 4) property panel.

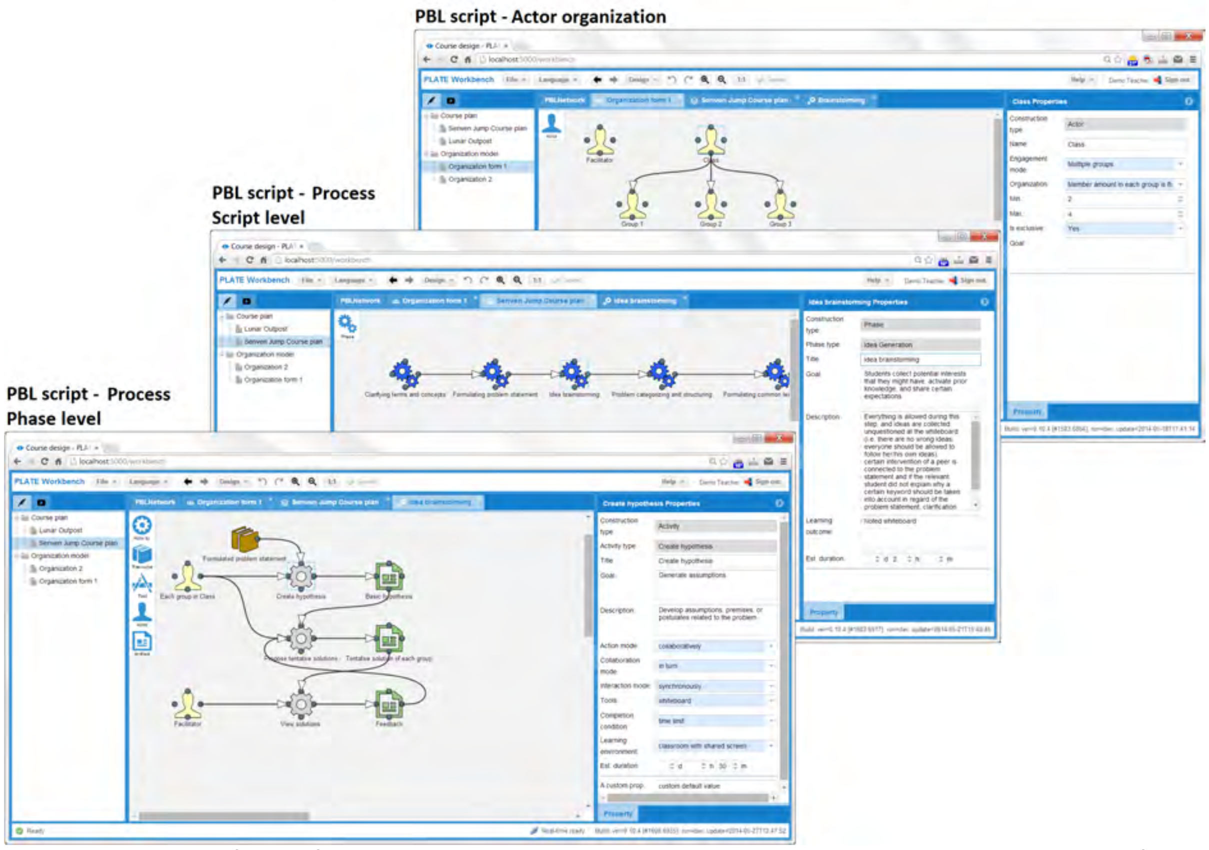
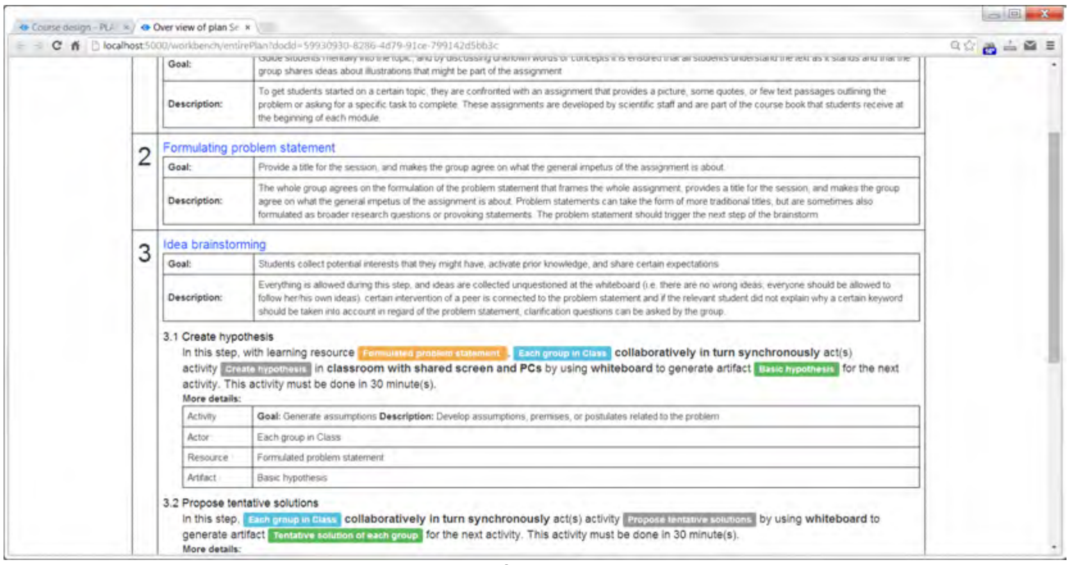

PBL script - Actor organization
PBL script - Process Script level
PBL script - Process Phase level

Figure 2. User interfaces of actor organization design, script level design and phase level design of PBL script in the PBL authoring tool.

Specifically, in the organization design, teacher can drag the *actor* meta-type icon from the meta-type pool and then drop it into the edit-space to create an actor node. Then the teacher will be asked to determine its type. This ensures that the node is appropriate for correct PBL process as well as can be understood by computer. According to the definition of PBL scripting language the type of actor could and only could be an *individual*, a *single group* or *multiple groups*. Teacher just needs to choose one to define it. For example in the organization design UI, an *individual* node is crated and named as *Facilitator*, a *multiple groups* node is crated and named as *Class* (in the property panel we can see the minimum participants of each group is 2, maximum is 4, and so further). Teacher can also create connections between them to define their relations. The organization can be shared by multiple PBL processes.

After an organization is defined, teacher can begin to define the core part of a PBL process: script level process and phase level process. Almost the same to the way of designing organization, teacher can also simply by dragging, dropping, choosing, editing properties, connecting, etc. to design these two levels of process. But compared to the organization design, there are 3 other important points need to be emphasized. 1) There are more meta-types which include *phase*, *activity*, *resource*, *tool* and *artifact* generated according to our language meta-model to the meta-type pool while in the organization design there is only *actor*. Actually, the process of creating these types (including *actor*) of nodes on the edit-space is the process of instantiating node types defined in the language meta-model to node instances. In other words, teacher just needs to instantiate the meta-types instead of designing from scratch by themselves. This mechanism ensures that the design of script is both flexible and responsible (Wang et al., 2014). As it is shown in the script level UI of figure 2, the node *Idea brainstorming* is based on the type *Idea Generation*. *Idea Generation* is not defined by teacher but by expert. So that when teacher designs the internal structure of the *Idea brainstorming* node (phase level design), all available node types and nodes' properties under this phase are only the appropriate types and properties for this phase according to the meta-model's specification. For example, when teacher wants to create an activity, the only options are *Offer conjecture*, *Create hypothesis*, *Propose tentative solutions*, etc. Or when teacher wants to create an artifact, only *Hypothesis*, *Solution*, etc. can be chosen. Inappropriate node definition will be avoided. 2) The types of actor node are the nodes defined in actor organization, but with corresponding extension. For example, since there is a *multiple groups* type node *Class* created, when teacher drags an actor into the edit-space, a list which contains *All members in Class*, *Each member in Class*, *All groups in Class* and *Each group in Class* will be generated for choosing. So in the phase level design UI, an *Each group in Class* actor node is there as an example. Connected this node, there is a *Create hypothesis* node. In property panel of this node, for example, teacher can choose *collaboratively* as the *Class*' *Action mode*, and let each group *in turn* perform the activity. Each group will *synchronously* use *whiteboard* to generate their *Basic hypothesis* within *30 minutes*. 3) Except there are fixed properties for each node in the script and phase design level, it is possible to add new custom properties into the property panel, because the property requirement in real world is always uncertain. As you can see there is an *A custom prop.* at the bottom line in the property panel in the PBL phase level design UI. It is achieved through extending our scripting language meta-model by experts, rendered in the script design UI, which is used to save, retrieve more information to get higher expression ability.


Course design - PLA
Over view of plan Se
localhost:5000/workbench/entirePlan?docId=59930930-8286-4d79-91ce-799142d5bb3c

Goal:	Guide students mentally into the topic, and by discussing unknown words or concepts it is ensured that all students understand the text as it stands and that the group shares ideas about illustrations that might be part of the assignment
Description:	To get students started on a certain topic, they are confronted with an assignment that provides a picture, some quotes, or few text passages outlining the problem or asking for a specific task to complete. These assignments are developed by scientific staff and are part of the course book that students receive at the beginning of each module.

2 Formulating problem statement

Goal:	Provide a title for the session, and makes the group agree on what the general impetus of the assignment is about.
Description:	The whole group agrees on the formulation of the problem statement that frames the whole assignment, provides a title for the session, and makes the group agree on what the general impetus of the assignment is about. Problem statements can take the form of more traditional titles, but are sometimes also formulated as broader research questions or provoking statements. The problem statement should trigger the next step of the brainstorm

3 Idea brainstorming

Goal:	Students collect potential interests that they might have, activate prior knowledge, and share certain expectations
Description:	Everything is allowed during this step, and ideas are collected unquestioned at the whiteboard (i.e. there are no wrong ideas; everyone should be allowed to follow her/his own ideas). certain intervention of a peer is connected to the problem statement and if the relevant student did not explain why a certain keyword should be taken into account in regard of the problem statement, clarification questions can be asked by the group.

3.1 Create hypothesis

In this step, with learning resource Formulated problem statement, Each group in Class **collaboratively in turn synchronously** act(s) activity Create hypothesis in **classroom with shared screen and PCs** by using **whiteboard** to generate artifact Basic hypothesis for the next activity. This activity must be done in 30 minute(s).

More details:

Activity	**Goal:** Generate assumptions **Description:** Develop assumptions, premises, or postulates related to the problem
Actor	Each group in Class
Resource	Formulated problem statement
Artifact	Basic hypothesis

3.2 Propose tentative solutions

In this step, Each group in Class **collaboratively in turn synchronously** act(s) activity Propose tentative solutions by using **whiteboard** to generate artifact Tentative solution of each group for the next activity. This activity must be done in 30 minute(s).

More details:

Figure 3. The textual output of the graphical designed PBL script.

When teacher finishes his or her design, a graphical PBL script can be exported as a complete textual script. Figure 3 shows the textual output of the example used in figure 2. Notice that if the teacher did not specify the *Goal* or *Description*, etc. the tool will automatically generate a default general definition based on the language meta-model and the node type chosen by the teacher at design-time. In this output, the top level contains the phases; then each phase contains activities (for saving space and as an example, there is no internal definition in phase 1 and 2). In phase 3, *Idea brainstorming*, we can see the activities inside. A summary generated by the tool for each activity according to which *actor*, *resource*, *artifact*, etc. it connects and what property values are set. For example the generated summary of activity 3.1 *Create hypothesis* is: "In this step, with learning resource *Formulated problem statement*, *Each group in Class* collaboratively in turn synchronously act(s) activity *Create hypothesis* in classroom with shared screen and PCs by using whiteboard to generate artifact *Basic hypothesis* for the next activity. This activity must be done in 30 minute(s). "

This proves that the tool has already understood what the teacher's script means. So interoperating the graphical script for other run-time learning systems, e.g. IMS-LD players, becomes possible.

4. An Approach of Supporting the Representation and Management of PBL Scripts

The previous section has given an initial impression of our approach. But to actually implement it, several technical problems need to be overcome: 1) How to support this kind of unified manner to represent actor organizations and the processes? Because the heights of the script document tree could be greater than 1 or 2 as the example. The degrees of the sub-trees could also be greater than 5, 7 or 9 as the example. 2) How to effectively manage (save, retrieve and compute) the graphical scripts? The saving/retrieval includes saving/retrieving all the nodes, saving/retrieving (custom) properties of every node, saving/retrieving all the relations between the nodes, saving/retrieving the multi-levels structure; the computing includes transforming graphical scripts to complete textual documents. This section will present a flexible system and data model approach to overcome these problems.

4.1 The Implementation Concept

Although the multi-levels structure of a PBL script is semi-structured, which has uncertain height and degree, it has a special hierarchical characteristic. That is, different level is for definition, same level is for relation. For example, the script level is the general process definition of a script divided by phases; the phase level is the detailed definition of each phase. In contrast, the script level defines the relations between phases; the phase level defines the relations among activities, actors, etc. Similar, if one wants to give more fine-grained definition for the *Class*, one can also define a lower level for it. Therefore, the edit-space can be designed to support the design of a whole PBL process iteratively. Each edit-space represents the definition of only one upper layer node. Each definition is a directed graph that consists of nodes and connections. Figure 4 illustrates this kind of iterative representation from two aspects. The edit-space 1 is for designing the phase's sequence for example. After a phase was defined it can be opened into a new edit-space. Similar to the first, in the edit-space 2, activities, resources, tools, actors, artifacts and their relations can be defined. If required, a third level of edit-space can be opened. Therefore a process script even if with infinite height and degree can be defined. Since manipulation requirement of each level is similar, the functionalities required for each edit-space are similar too.

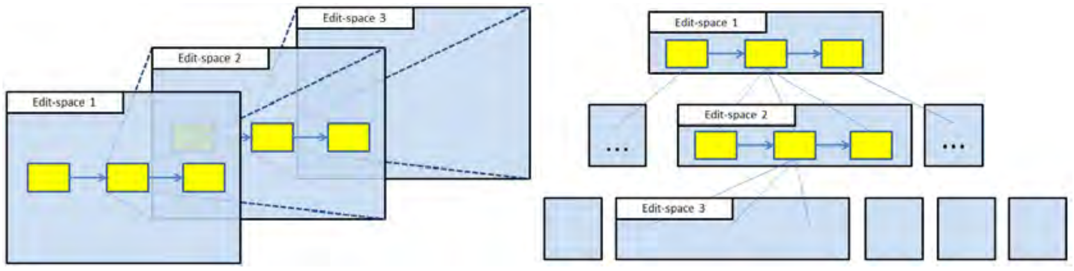


Figure 4. Iterative representation concept of a PBL process in multi-level edit-spaces

Actually, database can be also designed to save the definitions in each edit-space iteratively in order to save infinite height and degree of PBL scripts. Because there are only finite meta-types of nodes in the PBL scripting language, which only includes *phase*, *activity*, *resource*, *tool*, *actor* and *artifact*, and nodes in edit-space are just the instances of those meta-types, all nodes can be iteratively saved in their own meta-types of collections. Relation connections among nodes are seen as connection type instances which can be saved in a connection collection. Figure 5 illustrates this kind of storage approach. Each node has a *definition id* field as a pointer points to another edit-space; each edit-space has a *segment id* to let the nodes or connections know where they locate. Nodes and connections of different edit-spaces from different scripts are scattered stored in their own meta-types of collections.

The retrieval of a definition is just to find all the nodes and connections where their *segment id*s equal to the edit-space's *segment id*. Then the data can be directly sent to client side without any logical calculation. Basing on their graphic coordinate values all nodes and connects will be correctly rendered on edit-space. Because the nodes and connections as minimum units are stored separately, it shows several benefits: 1) It is very effective to add, remove, update and find node whenever where it is in one or many PBL scripts. When teacher edits a node or connection, any change can be saved without involving any other nodes or connections. So high computation cost in updating script is avoided. 2) It is very easy to handle the change of process structure. Because the node definitions are not really connected to each other, the change of structure only needs to change the pointer value. This benefit make it is possible to effectively reuse and share any levels of sub-trees of any script. When teacher wants to view the complete script, all nodes and connections can be easily retrieved because all of them whom belongs one complete script have a same *script id*. The only additional thing is that before a complete script is sent back to client, a semantic engine will organize them into a correct order.

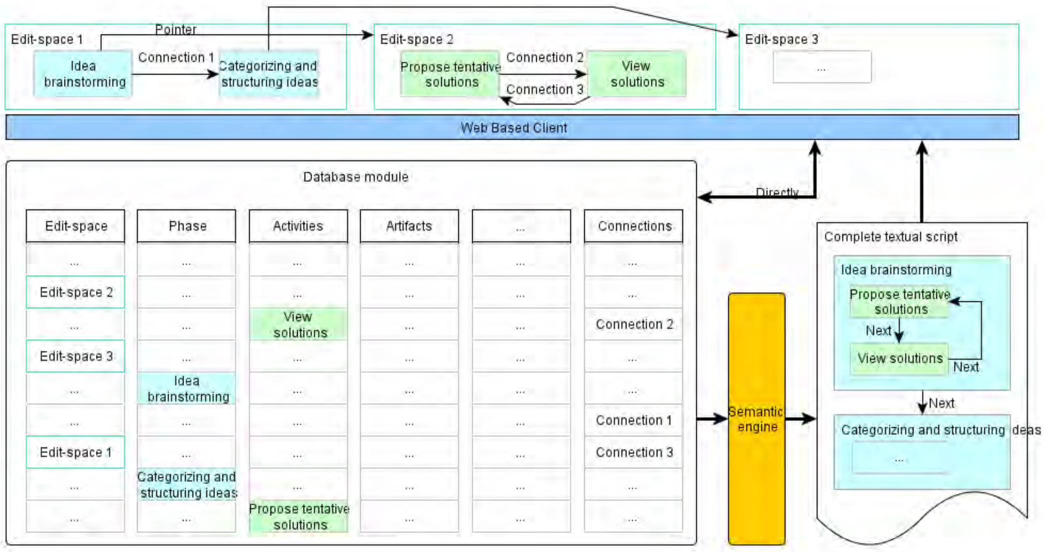


Figure 5. The storage concept of the PBL script

4.2 *Combined Use of Semi-structured Data and Unified Data Format*

The previous section has elaborated the implementation concept of representing and managing the semi-structured script data, but to technically implement it is another challenge. The following facts must be considered: 1) Not only the structure itself of PBL script is semi-structured, but also the node data itself is semi-structured. For example, as mentioned before, node can have custom fields. The data type of a field could be string, array or others. Figure 6 shows this characteristic. In comparison to activity *Propose tentative solutions*, the custom fields of activity *View solutions* may be more complicated, if it needs an array to store the values for a drop-down list of a combo box. 2) As a web-based application, to maximally avoid browser compatibility issues, we need to use pure JavaScript to develop the rich graphical client.

Because the Node.js can be chosen as our web server, we can use JavaScript to program both client side and server side. Figure 7 shows the system architecture from the perspective of data handling. Since the client side and sever side of the system are both built by using JavaScript and JSON is the object notation of JavaScript; also because we can use MongoDB as our system database and the data persistence format in MongoDB is JSON (or BSON[1]), the data interchange format inside the whole system can be unified as JSON object. As we know, JSON is an ideal data-interchange language for representing semi-structured data. Therefore the node/connection objects in the system could be simply as JSON objects. Consequently, on the one hand, client side and server side can communicate each other directly through JSON object (after serialization). In client side, nodes and connections can be directly rendered to edit-space or sent back to server side without any format transformation. Also without any transformation, server side can directly process the nodes or connections from client side. On the other hand, data from client side can be stored inside the database directly. Even if there are new

[1] Short for Binary JSON, is a binary-encoded serialization of JSON-like documents.

custom attributes, such as string or array fields, which are customized added, client and server side JavaScript interpreter and MongoDB can still handle and store them natively.

Together with using pure JavaScript for the client UI, Node.js as the web server and MongoDB for the data persistence, JSON becomes a natural choice as a unified data format in our whole system. Combined with the implementation concept for handling the semi-structured process descriptions, all the requirements for the representation and management of PBL scripts are met.

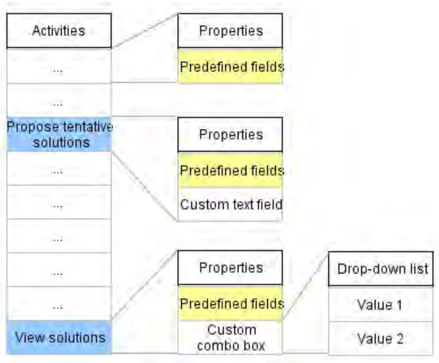


Figure 6. Custom property in node

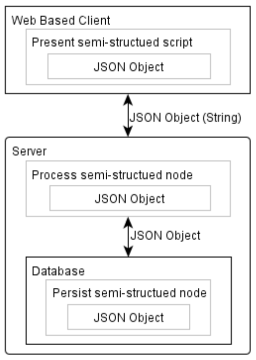


Figure 7. System architecture from the perspective of data handling

5. Related Work

Generally speaking, currently there are two kinds of implementation that can flexibly support PBL design: IMS-LD authoring tools and the LAMS (Dalziel, 2003). For IMS-LD authoring tools, there are Reload (Reload 2005), MOT+ (Paquette, et. al, 2006), ASK-LDT (Karampiperis & Sampson, 2005) CopperAuthor (CopperAuthor 2005) and CoSMoS (Miao, 2005). These tools are very flexible to represent and support the design of different learning process models and output the models as Unit of Learning (UoL) packages. The packages can be interpreted by IMS-LD run-time applications, such as CopperCore player (Martens & Vogten, 2005) and SLED (McAndrew, Nadolski & Little 2005). All these tools are general learning design tools; they have the capability of supporting PBL design. For the LAMS, a study has shown that it also can be used for PBL design (Richards & Cameron, 2008).

However, all of them have some shortcomings in term of supporting PBL design. From the perspective of supporting visual learning design, IMS-LD authoring tools and LAMS misses the capability of facilitating teachers in developing sound PBL process since they are too general and not PBL domain specific. From the perspective of utilizing Web technologies, these tools came about by adopting traditional software development concept, most of them are desktop applications. As we know, desktop applications have high maintenance cost, are not anywhere available without pre-installation, are not cross-platform (if they are not on the top of Java) and so further. From the perspective of data management approach, existing applications store learning design artifacts either by using traditional relational database or directly as XML files. Relational databases are good at data storage and querying, but they cannot manage this kind of semi-structured data well since they are relational and not schema free. XML file is a kind of ideal media for storing this kind of semi-structured data; one drawback is that it is not ideal for data manipulation, such as partial update, search, etc. which will lead to considerable computational overhead in server side. Although there are some combination solutions to manage XML documents through relational database, XML queries are still inefficient (Shanmugasundaram, et. al. 2008).

6. Summary and Future Work

PBL is a kind of high collaborative learning process with a distinctive pedagogy, therefore there are difficulties in designing the process script, which include designing actor organization and designing learning process. Thus an extended CSCL scripting language, the PBL scripting language, is developed.

To easy the applying of the language in Web 2.0 ear, we developed a web-based PBL authoring tool to provide a rich and intuitive representation for helping design any kind of PBL scripts of any kind of models. Our former pilot study (Miao et al., 2013) has shown that the difficulties to design and delivery pedagogically high-quality, human readable, reusable, sharable and computer-executable, online or blended PBL process are reduced as intended. Our tool has been implemented based on the premise that there are two common dimensions of semi-structured characteristics inside PBL scripts. One is from the point of view of the scripts; the other is from the point of view of a single node/connection inside scripts. From the first dimension, if we see a script document tree from root to leaf, the sibling nodes and relations are the definitions of its upper level parent node. That is, nodes (connections) are used to define node; if we only focus on one sibling level, horizontally it is just about the definition of the relations among the sibling nodes. The sibling nodes do not define each other. So by iteratively doing these two things, any PBL script can be defined, regardless how complicated an organization or a learning process is. From the second dimension, we use JSON as a unified data process and storage format. Since JSON format is designed to represent semi-structured data, the uncertainty of node's or connection's structure can be processed. In order to effectively implement these two dimensions of design as a web-based application, we use JavaScript to program both client side and server side. In case of storing and retrieving the JSON data, we use MongoDB as database, so that all the semi-structured characteristics of the two dimensions can be effectively handled. At last we also discussed some other related work. In term of user interface and data management, other work more or less has their drawbacks. Therefore we conclude that a combined use of semi-structured data management and a unified data format appears to be a promising approach to effectively and efficiently support the representation and management of PBL scripts.

This paper illustrates an approach to design and implement a web based authoring tool to help teachers represent and manage their PBL mental scripts as graphical scripts. And then the graphical scripts can be transformed into textual scripts or learning systems executable scripts. Our future work includes two directions: one is to make these transformation steps automatically. That is, extract process directly from existing semi-structured textural scripts, and then store them in system, represent them by our graphical user interface and transform them to other learning systems; the other is to develop our own specific PBL run-time learning environment, so that we can provide an integrated way to test and run the PBL scripts for both teachers and learners.

Acknowledgement

This paper was made possible by NPRP grant # NPRP 5-051-1-015 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Ali, Z., & Samaka, M. (2013). “ePBL: Design and Implementation of a Problem-based Learning Environment,” Proc. of 2013 IEEE Global Engineering Education Conference (EDUCON), 1209-1216, Berlin, Germany.

Barrows, H. S. (1996). Problem-based learning in medicine and beyond. In L. Wilkerson & W. H. Gijselaers (Eds.), *New directions for teaching and learning: Vol. 68. Bringing problem-based learning to higher education: Theory and practice* (pp. 3-13). San Francisco: Jossey-Bass.

Boud, D., & Feletti, G. (1998). The challenge of problem-based learning. London: Kogan Page.

CopperAuthor (2005). CopperAuthor Learning Design editor, retrieved October 27, 2005 from http://sourceforge.net/projects/copperauthor/.

Dalziel, J.R. (2003). Implementing learning design: the Learning Activity Management System (LAMS), Interact, Integrate, Impact, 593–6, *Proceedings of the ASCILITE conference*. Available online at: http://www.ascilite.org.au/conferences/adelaide03/docs/pdf/593.pdf, last accessed 25 May 2014.

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.

Dirckinck-Holmfeld, L. (2002). Designing Virtual Learning Environments Based on Problem Oriented Project Pedagogy. In L. &. Dirckinck-Holmfeld & B. Fibiger (Eds.), *Learning in Virtual Environments* (pp. 31-54). Frederiksberg C: Samfundslitteratur Press.

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., & 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., & 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.

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

Kaldoudi, E., Bamidis, P., Papaioakeim, M., Vargemezis, V. (2008). Problem-Based Learning via Web 2.0 Technologies. *21st IEEE International Symposium on Computer-Based Medical Systems*, pp.391-396.

Karampiperis, P., & Sampson, D. (2005). Designing learning services for open learning systems utilizing Learning Design. In Uskov, V. (Ed.) *Proceedings of the 4th IASTED International Conference on Web-based Education*, Grindelwald, Switzerland: ACTA Press, 279-284.

Martens, H., & Vogten, H. (2005). A reference implementation of a Learning Design engine. In: Koper, R. & Tattersall, C., *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training* (pp. 91-108). Berlin-Heidelberg: Springer Verlag.

Maurer, H., & Neuhold, C. (2012). "Ways of Knowing, Ways of Learning". Paper prepared for the Higher Education Academy Social Science Conference, 28 - 29 May, 2012, Liverpool, UK.

McAndrew, P., Nadolski, R., & Little, A. (2005). Developing an approach for Learning Design Players. *Journal of Interactive Media in Education,* 2005/14. Retrieved November 21, 2006, from http://www-jime.open.ac.uk/2005/14/.

McLoughlin, M., & Davrill, A. (2007) Peeling back the layers of learning: A classroom model for problem-based learning, in *Nurse Education Today*, 27(4), May 2007, Pages 271-277.

Miao, Y. (2005). CoSMoS: Facilitating Learning Designers to Author Units of Learning Using LD. In: *Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences*, *Proceedings of the 13th International Conference on Computers in Education*, 275-282, Singapore, IOS Press.

Miao, Y., Harrer, H., Hoeksema, K., & Hoppe, U. (2007b) "Modelling CSCL Scripts: A Reflection on Learning Design Approaches". a book chapter in Fischer, F., Kollar, I., Mandl, H., & Haake, J.M. (Eds.) *Scripting Computer-Supported Collaborative Learning: Cognitive, Computational and Educational Perspectives,* 117-135, Springer.

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.

Miao, Y., Holst, S., Haake, J.M., & Steinmetz, R. (2000b). PBL-protocols: Guiding and Controlling Problem Based Learning Processes in Virtual Learning Environments. In: *Proceedings of the Fourth International Conference on the Learning Sciences* (ICLS' 2000), pp. 232-237. June 14-17, 2000, Ann Arbor, U.S.A.

Miao, Y., Samaka, M., & Impagliazzo. (2013). Facilitating Teachers in Developing Online PBL Courses, *Proc. IEEE TALE*, Bali, Kuta, Indonesia, August 26-29 2013, pp. 454-459.

Mills, D. (2006). Problem-based learning. *The Higher Education Academy, Sociology, Anthropology, Politics (C-SAP).*

Paquette, G., Léonard, M., Ludgren-Cayrol, K., Mihaila, S., & Gareau, D. (2006). Learning Design based on Graphical Modelling. *Educational Technology & Society*, 9(1), p. 97-112.

Reload (2005). Reusable e-Learning Object Authoring & Delivery Project, retrieved on October 27, 2005 from http://www.reload.ac.uk/.

Richards, D., & Cameron, L. (2008). Applying Learning Design concepts to problem-based learning In L. Cameron & J. Dalziel (Eds), *Proceedings of the 3rd International LAMS & Learning Design Conference 2008: Perspectives on Learning Design.* (p.p. 87-96). 5th December 2008, Sydney: LAMS Foundation.

Savery, J. R. (2006). Overview of problem-based learning: Definitions and distinctions. *Interdisciplinary Journal of Problem-based Learning,* 1, 9-20.

Shanmugasundaram, J., Tufte, K., He, G., Zhang, C., DeWitt, D., & Naughton, J. (2008). Relational databases for querying xml documents: Limitations and opportunities.

Wang, D., Miao, Y., Hoppe, U. and Samaka, M., A Domain-specific Modeling Language Approach to Support Various Forms of Online PBL, *In Proceedings of the 14th IEEE International Conference on Advanced Learning Technologies - ICALT2014*, July 7-9, 2014, Athens, Greece.

Woods, D. R. (1996). Problem-based learning for large classes in engineering education. *Bringing problem-based learning to higher education.* L. Wilkerson and H. Gijselaers. San Francisco, CA, Jossey-Bass: 91-99.