A Model-driven Approach to Developing a Web-based Environment to Support Problem-based Learning

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Informatik und Angewandte Kognitionswissenschaft
der Universität Duisburg-Essen
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von
Disi Wang
aus
Hubei (China)

1. Gutachter & Betreuer: Prof. Dr. Heinz Ulrich Hoppe
2. Gutachter: Prof. Dr. Andreas Harrer
Betreuer: Dr. Yongwu Miao
Tag der mündlichen Prüfung: 11.10.2018

Contents

List of Figures

Preface

The work presented in this thesis was conducted in the context of the project **P**roblem-**b**ased **L**earning **A**uthoring and **T**ransformation **E**nvironment, **PLATE** (2013-2016, QNRF). This project sought to provide students with an innovative technology-enhanced learning that helped them develop the knowledge, skills, and character more effectively and efficiently through facilitating teachers to design, represent, understand, communicate, customize, reuse, transform and implement online or blended Problem-based Learning (PBL) courses. Under this vision, various sub-objectives needed to be achieved. The general objectives included: develop a high-level scripting language; develop a new generation of a learning design tool based on the scripting language, enable this tool to generate computer-manipulatable PBL-infused scripts; develop a runtime tool and extent the functionality of other systems in order to execute the scripts for students as for courses.

Supported by this project, as the principal system developer and the system architect, the author worked in the research team was responsible for both finding a theory-based way to help teachers and students perform computer supported problem-based learning, and reviewing state of art technologies in search of an appropriate technical way to design and implement the system. Additionally, since this project planed to develop a PBL scripting language, the system was also required to be able to facilitate the theoretical research work in the domain of the PBL ontology, which means the author's work not only aimed at facilitating problem-based learning for teachers and students but also at facilitating the ontology research for PBL domain experts.

In this context, I did an in-depth review of the PBL pedagogy, Learning Design (LD), existing tools or systems for PBL, domain modeling, Domain Specific Language (DSL), Model-driven Development (MDD), Model-driven Engineering (MDE), Model-driven Architecture (MDA) etc. and then systematically formed a set of model-driven technical solutions, combined them to make use of contempo-

rary Information and Communication Technology (ICT), and finally achieved the goal of supporting both the PBL research and practice work of the project.

Abstract

Through studying the Problem-based Learning (PBL) pedagogy and Learning Design technology, summarizing the insufficiency of existing PBL applicable tools or systems, analyzing the requirements in technology-enhanced PBL research and practice, and utilizing the state of art Web technologies, this thesis presents a systematic work of researching, designing, and realizing a set of tools to provide a relatively complete solution that to a certain extent achieved the goal of facilitating PBL researchers and practitioners to design, represent, understand, communicate, customize, reuse, transform, and execute online or blended problem-based learning processes in an effective, efficient and flexible manner. This tool set was named as Web-based **I**ntegrated **D**esign and **E**xecution **E**nvironment for **P**BL, or shortly referred to as **IDEE4P**.

IDEE4P provides a higher availability and an intuitive user interface (UI) to PBL researchers and PBL education practice participants by utilizing the state of art Web technologies. It enables PBL pedagogy researchers to refine, reconstruct, and redefine PBL script language flexibly in order to achieve an important objective—supporting a general expressiveness for problem-based Learning Design—and, consequently, achieves the goal of facilitating PBL educational practitioners to represent and implement different forms of PBL based on their specific PBL implementation situations. Therefore, first, IDEE4P provides PBL pedagogy researchers a PBL scripting language editing tool that easily engages them in the language development; second, it provides PBL designers a PBL process authoring tool that enables them to easily perform the computer interpretable problem-based Learning Design; third, it provides PBL tutors a runtime tool that allows them to carry out learning activities for students based on the designed PBL process models in a online manner.

The foundation of designing and developing IDEE4P is to apply a model-driven approach by referencing the Model-Driven Development (MDD) methodology. In

this approach, a design-evaluation-improvement cycle is introduced in order to adapt the complexity and instability of the PBL research and practice. The core of this approach is to abstract a metamodel from PBL pedagogy. This metamodel controls the system to help people construct models of PBL processes appropriately and controls the transformation process to enable the models executable by computers servicing the real teaching-learning activity interactions. The metamodel abstraction is derived based on the modeling theory after analyzing the characteristics of existing mainstream PBL patterns and frameworks. This analysis work is based on earlier suggestions by Miao et al. (2005); Wang et al. (2014a). Based on the analysis, a set of constituent elements was developed for the metamodel. The element set was referred to as metametamodel in our system, which can be also seen as the fundamental abstraction of the PBL pedagogy.

To evaluate the usability, functionality and the relative new modeling concept provided by the system, we organized a comprehensive pilot study. We also conducted several other case studies, pilot studies, and workshops that were mostly from the pedagogy point of view on the IDEE4P system. This thesis will present two pilot studies selected from those two evaluation branches. From the results of the evaluation work, we concluded that a model-driven approach based, state of art information and communication technology enhanced, online PBL environment appears to be a promising effective and efficient support for PBL research and practice in facilitating the Learning Design and the corresponding learning implementation work.

There are still some important evaluations and functionalities that needed to be conducted and improved. These points will be discussed as the future work as one part of the outlook. The author also applied the same approach to design and develop another workflow-based learning system. The feedback of the system development and early test results showed that it seems that the model-drive and the engineering methodology presented in this thesis was also suitable for this kind of use case. However, formal evaluation was also needed, which will be the other point in the outlook.

Acknowledgment

This thesis was made possible by NPRP grant # NPRP 5-051-1-015 from the Qatar National Research Fund (a member of the Qatar Foundation).

This thesis was also made possible by the great help from Dr. Y. Miao and Prof. H. Ulrich Hoppe.

Part I

Theoretical Research

Chapter 1

Introduction

1.1 Background - 21st Century Competencies

All along, an uncountable number of educational researchers and practitioners were committed to helping students develop their knowledge and skills in a more efficient way, with the hope of guiding students to gain personal success more easily and provide output in the workforce more productively. Particularly in the past decades, scholars consistently pointed out that our education systems must keep pace with our changing world since the workforce of nowadays world is changing from serving an industrial model in the industrial age of the 19th century, in which jobs were mostly low skilled and employees were only required to follow basic procedures designed by others, to serving a rapidly transforming, technology-driven, and interconnected globalized knowledge economy of the 21st century (Barron and Darling-Hammond, 2008).

Therefore, our education systems are forced to provide better or new pedagogy strategies to help students meet this new situation. These strategies need to enable students to learn on demand from desire, particularly emphasizing more natural engagement in critical and creative thinking (Trilling and Fadel, 2009) and familiarize students with communicating and collaborating, information collecting, ideas researching, and synthesizing (Barron and Darling-Hammond, 2008), as well as consequently develop the capability of solving ill-structured, complex, and unknown problems. This capability is generally called 21st century competencies[1]

[1]21st century competencies is just one of the widely used terms for this concept. Other synonymous terms, such as 21st century skills, deeper learning, global competencies, student-

(Pellegrino et al., 2013).

The 21st century competencies is not yet 100% unified (Rychen and Salganik, 2001; Dede, 2010; Center, 2010; Griffin et al., 2012), various frameworks keep evolving and developing from different organizations, such as the Assessment and Teaching of 21st Century Skills (ATC21S), European Commission, National Academy of Sciences (of United States), Organisation for Economic Co-operation and Development (OECD), the Partnership for 21st Century Learning (P21), U.S. Department of Labor, etc[2].

For instance, the Partnership for 21st Century Learning, addresses the competencies as the following (P21, 2007):

- **Creative and Innovation**: thinking creatively, working creatively, and implementing innovations.
- **Critical Thinking and Problem Solving**: reasoning effectively, using systematic thinking, making judgments and decisions, and solving problems.
- **Communication and Collaboration**: communicating clearly and collaborating with others.
- **Information and Media Literacy**: accessing and evaluating information, using and managing information, analyzing media, and creating media products.
- **Information, Communications, and Technology (ICT) Literacy**: applying technology effectively.
- **Flexibility and Adaptability**: adapting to change and being flexible.
- **Initiative and Self-direction**: managing goals and time, working independently, and being self-directed Learners.
- **Social and Cross-cultural Skills**: interacting effectively with others and working effectively in diverse teams.

centered learning, etc., are also used by different organizations or research groups.

[2]The terms *skill* and *competency* represent distinct meanings. According to the Terminology of European Education and Training Policy (Europ., 2014), competence (the uncountable form of competency) is the "ability to use knowledge, skills and personal, social and/or methodological abilities, in work or study situations and in professional and personal development," while skill is the "ability to apply knowledge and use know-how to complete tasks and solve problems." However, in the context of describing the 21st century competencies, the term *skill* and *competency* or *skills* and *competencies* are sometimes interchangeable, which means the term *21st century competencies* indicates the same thing as the *21st century skills*

- **Productivity and Accountability**: managing projects and producing results.
- **Leadership and Responsibility**: guiding and leading and being responsible to others.

Although different organizations or groups develop different definitions of competencies, the major content is similar, meaning that these frameworks appear to be in broad agreement.

To meet this educational requirement introduced above, the traditional education approach — which allows teachers to be the only ones imparting knowledge to the students and see them as simple receptors — is not sufficient. In traditional classrooms, teachers narrow tasks and use a single standard to assess the achievement of each student, and teachers treat every student equally and teach them the same things. According to Peter F. Oliva (2013), traditional education is organized as a manner of textbook-driven, teacher-centered, and student memorizing facts. In this way, students passively receive the information, causing difficulty in establishing a good interaction between the teachers and students. Knowledge is discretely divided into different curricula, thus students feel the knowledge irrelevant to their daily lives. Therefore, students do not deeply involve themselves in learning activities, which doesn't help them develop critical thinking (Bransford et al., 1999) etc.

1.2 PBL and 21st Century Competencies

If we want students to learn more deeply, we need to encourage them to engage in solving more authentic real world problems (Newmann et al., 1996). They need to be exposed to genuine problems. Darling-Hammond et al. (2015) believed that the new education approach should also feature the following points: support inquiry-based learning, provide collaborative learning, teach students how to learn rather than only what to learn. Therefore, Problem-based Learning (PBL) was considered as an innovative alternative approach to the traditional approach (Evensen and Hmelo-Silver, 2000; Schmidt and Moust, 2000).

PBL is a learner-centered education approach. Not like the traditional education approach, where the teacher is the center and only disciplinary knowledge will be exposed, in PBL realistic case problems are used as stimuli and multiple solutions

are expected from students rather than a predefined "right" one. Teachers are instead facilitators or coaches, students are instead initiators in an open context, free to choose the information basing on their individual understanding, positively solving various problems, and building different learning paths. In this way, students are driven to actively, collaboratively gain knowledge and develop their critical thinking, problem-solving, team-work, and self-directed learning skills (Boud and Feletti, 1997b; Hmelo-Silver and Eberbach, 2012; Barrows, 1986; Hmelo-Silver et al., 2007). Barron and Darling-Hammond (2008) argued that PBL was able to meet most of the educational requirements for the increasingly complex life and career environment in the 21st century.

PBL was initially developed for medical education. Early in 1980, Barrows, in his book "*Problem-Based Learning: An Approach to Medical Education*," originally figured out that educators needed to develop a new educational approach that emphasizes reasoning and understanding rather than rote learning and memorizing. He claimed that this kind of learning approach was "a rigorous, structured approach to learning that is tailor-made for medical education and based on considerable experience and research," and that the approach was "the learning which results from the process of working towards the understanding or resolution of a problem" (Barrows et al., 1980). Later, researchers found that this approach was one of the most significant innovations in education for professions (Boud, 1985).

After a period of development of this approach, scholars further figured out that PBL is not static. It is changed "from the approach which arose from the unique context of medical education. Nevertheless, those original ideas were sufficiently robust [in] that they have provided the foundations for many others elsewhere" (Boud and Feletti, 1997b). This statement meant that with the continuous expansion of the study scope and the innovative extension of the boundaries, more and more scholars found that problem-based learning was not only good for medical education, but also for other professional areas. For instance, it started to make inroads into engineering and science education (Woods, 1996; Johnson, 1999; Mills et al., 2003; Yadav et al., 2011). Nowadays, PBL has been successfully used in various domains or in large classes where student groups were even without a tutor (Woods, 2006).

1.3 Motivation

Nevertheless, despite PBL having so many advantages and benefits and being widely recognized, it is still not popular, or in other words, it has not been widely applied yet. One important reason for this kind of divergence phenomenon is that an appropriate implementation of PBL is not easy for current teachers (Boud and Feletti, 1997a; Kirschner et al., 2006). Early in 1997, when Boud and Feletti (1997b) wrote the second edition of their famous book *The Challenge of Problem-based Learning*, they had already figured out that "...there have also been misapplications and misconceptions of PBL, some of which relate to the particular features of the approach, others which involve the challenges associated with instituting major educational change." More specifically those misapplications and misconceptions indicated that there was the following (Boud and Feletti, 1997b):

- Confusion concerning PBL as an approach to curriculum design with the problem-solving teaching;
- Adoption of the PBL proposal without sufficient commitment of staff at all levels;
- Lack of research and development on the nature and type of problems to be used;
- Insufficient investment in the design, preparation and ongoing renewal of learning resources. This was commonly associated with a lack of recognition of the additional resources of the start-up phase of a problem-based curriculum;
- Establishment of small elements of PBL in a context that rewarded students for the kind of behavior favored in a traditional lecture-plus-examination environment;
- Insufficient concern with staff induction and development, particularly for those staff who were not part of the original team that developed the program;
- Inappropriate assessment methods that did not match the learning outcomes sought in problem-based programs; and
- Evaluation strategies that did not focus on the key learning issues and were implemented and acted upon too late.

In fact, for a long time, teachers found it very easy to make these mistakes mentioned above because they were not PBL experts or cognitive scientists with the expertise to transform lecture driven courses into problem-driven courses. They

constructed PBL courses based on their personal education experience and implicit decisions. They were well-versed in teaching and lecturing; while it was reasonable to have a difficulty in changing their role to facilitator, they need to aim at guiding students rather than just giving answers (Ertmer and Simons, 2006). However, students did not adopt this approach smoothly. They also needed time to adjust themselves to the learning style, to move past the traditional way to a self-directed, inquiry and collaborative manner.

Moreover, there was no agreement on the best practice to guide the implementation of PBL. Depending on different emphasis concerns, PBL could be conducted in a number of ways based on various practice experiences such as the McMaster PBL model (Woods, 1996), the Maastricht "Seven jumps" model (Barrows, 1996), Seymour's "five-stage" model (Seymour, 2010), SALFORD model (McLoughlin and Darvill, 2007), and etc. This situation confused teachers more because they needed to decide at the very beginning which one was the best for their particular situation.

Today is an age of universally making use of Information Communication Technology (ICT). Recently, emerging pieces of evidence showed that making use of technological innovations in education transformed the learning style (Levy and Murnane, 2007; Sharples et al., 2015). To solve the problem summarized above, utilizing ICT to facilitate PBL was the right way. However, from the point of the view of applying ICT, many teachers lacked a basic understanding of the potential affordances from technologies since they were more likely to present their ideas in their natural language and carry the course (lesson plan / learning scenario) on paper. Conversely, from the point of the view of the support that ICT gave, there was no good tool that could help PBL for a long period of time.

Since the insufficiency of technology enhanced PBL was easy to observe, some technology enhanced solutions have been developed to support PBL (Kaldoudi et al., 2008). For example, STELLAR supported conducting the PBL process through a nine-step model; ePBL was based on the McMaster PBL model. These applications were easy to use because each had a well-designed PBL pattern inside. While there were also some other applications, such as LAMS and ISM-LD related tools, which were very flexible in helping teachers carry out different patterns of PBL, there were still some serious problems. For those tools with well-designed PBL patterns inside, they were too rigid to use. This meant that teachers could not configure the sequence of learning activities or customize certain activity units

for their particular practices or purposes. For example, if teachers wanted to apply Seymour's five-stage model, these tools would not be able to facilitate this specific model. For those tools which had flexible PBL pattern definition features, they were inadequate in helping teachers efficiently conduct sound PBL processes, especially when teachers did not have enough understanding of PBL pedagogy. With these tools, when designing a PBL process, teachers needed to figure out which activities were appropriate for which phase or what kinds of artifacts should be provided as temporary or final learning outcomes for certain phases. A More detailed analysis will be elaborated later in Section 3.

As a result, because of the reasons mentioned above, with current PBL practice, teachers still tend to embed their learning process in their practice, and the process design ideas tended to be implicit. This fact led to PBL processes being mostly implemented only based on the social protocol and the manual configuration of various application tools, as well as the manual management of learning resources and (non-) digital learning artifacts. More specifically, in most situations, PBL teachers need to use a mixture of different tools, such as email, discussion forum, blog, wiki, Skype, etc. to facilitate PBL. This is far less than effective and efficient. When using PBL this way, teachers still need to have paper-based course plans before courses. They have to carefully and timely manually coordinate learning processes with those tools according to their plans as they implement their courses. Additionally, students have to manage artifacts in different forms and shift their learning activities among the various workspaces even in a single learning section. Consequently, directing a high-quality technology-supported PBL course, combining the benefits of the emerging ICTs, and keeping pace with the requirements of 21st century competencies, became a very complicated and ineffective work. Especially regarding utilizing ICT, the non-integrated technology enhanced learning to some extent even goes backward.

In summary, by now, no one generic system can systematically enable PBL practitioners to develop and deliver online or blended PBL courses in an easy, cost-effective, flexible, interoperable, and reusable way. We can assert that if there are no new innovative technology enhanced supports to PBL, to achieve the full power of PBL it will stay in a work cycle of research, application, assessment and redesign for years.

1.4 Problem Statement

Facing all the findings elaborated above, the project named, **P**roblem-based **L**earning **A**uthoring and **T**ransformation **E**nvironment, shortly called **PLATE**, was founded. This project brought learning scientists, PBL experts, and computer scientists together to conduct research developing an innovative approach and an online system. The vision of this project was to help teachers easily, flexibly, and responsively implement a wide range of models of online or blended PBL; help teachers to make the implicit PBL design process explicit, help teachers to improve their design quality, and guide teachers to represent traditionally informal descriptions as formal models that can be understood by computer for scaffolding and orchestrating ICT enhanced PBL. Specifically, according to the proposal of the project, the goal of this project was composed of the following three aspects:

1. Develop a high-level scripting language and a new generation of learning design environment that flexibly and efficiently supports the development of PBL-infused modules and course;
2. Create a run-time environment that supports the execution of PBL models and that scaffolds and orchestrates PBL activities to help students to transition into new roles and to learn within PBL contexts;
3. Test the extent in which the PLATE system offers a flexible and adaptive environment for both teachers and students learning with a PBL approach under different conditions.

Under these aspects, the associated objectives included:

- Develop a PBL scripting language to enable the representation of a wide range of PBL models, and investigate the expressiveness of that language;
- Generate two language-compatible authoring tools to support and mentor PBL researchers and practitioners in specifying, understanding, communicating, and customizing various PBL models and their variations, and evaluate the effectiveness and efficiency of using the authoring tools;
- Implement the PLATE run-time environment by adopting a model-driven approach and by extending Moodle and IMS LD platforms, and evaluating the feasibility and usability of the runtime environment;
- Integrate a repository in the authoring environment with a set of PBL models and process segments to enable the storage, retrieval, comment, rate, and

reuse of PBL scripts.

Thus, as a research assistant and the principal system developer of this project, I was confronted with the following questions:

- How do we design and develop the PBL authoring tool to support the different characters of a wide range of PBL models while the tool also provided service for the PBL runtime and the Moodle extension?
- As we can not expect a teacher to define a PBL process - such a kind of complicated collaborative learning process, with one-time success. In fact, a PBL process becoming mature always needs many design-evaluation-improvement cycles. Therefore how do we design these cycles to run easier for teachers?
- How do we help project researchers develop the PBL scripting language while as the same time making the new developed scripting language rapidly affect the manner of PBL process model authoring?
- Similar to defining a PBL process for teachers, we cannot also expect experts to define the PBL scripting language to be mature enough with one-time success. The development work of the PBL scripting language also needs multiple design-evaluation-improvement cycles. Therefore do we design these cycles run easier for PBL experts?
- How do we apply the model-driven approach to the PBL scripting development, the PBL process authoring and the PBL process runtime execution systematically?
- How do we design and store PBL script[3] data in order to achieve the requirement of the storing, retrieving, sharing, commenting, rating, and reusing features?

In summary, from the PLATE project perspective, the work aimed at helping teachers and their students perform PBL. However, from my perspective, the work was not only just to develop the tools helping PBL teachers and their students to conduct their ICT enhanced problem-based learning activities easier, but to also, which is important, find an appropriate way to design the system so that it could also facilitate the theoretical research of the project; or even the general research of the PBL pedagogy area since the system must also support the development of a PBL scripting language. In other words, my work aimed at both helping PBL teachers and students and supporting PBL researchers.

[3]From the perspective of computer science, we called a PBL course described through the PBL authoring tool as a PBL script or a PBL model or a PBL process.

1.5 Structure of the Thesis

Various challenges needed to be overcome to achieve the system goal as described above. In this thesis, I will present an in-depth review of the PBL pedagogy, Learning Design (LD), existing tools or systems for PBL, Domain Specific Language (DSL), Model-Driven Development (MDD), Model-Driven Engineering (MDE), Model-Driven Architecture (MDA), etc. and systematically elaborate a set of model-driven technical solutions to answer the questions listed above. In comparison with other tools and technical approaches, we concluded that making use of a model-driven approach combined with the contemporary ICT to build the whole system would be a promising approach to effectively and efficiently support the PBL researching, authoring, delivery, and execution from scripting language development to process design and finally to course running.

This thesis consists of the following chapters:

1. **Introduction**, the current chapter, introduces a general background of my work and introduces the motivation and the problem statement. Tt also provides the outline of this thesis.

2. **Theoretical Background** provides an explicit theoretical foundation of technology enhanced PBL. The topics include the principle of learning, more details about PBL, and what is **L**earning **D**esign.

3. **Applicable Tools** summarizes the current achievement and insufficiency of the technical supports for PBL in reviewing the features of PBL applicable tools or systems.

4. **Problem, Basic Idea, Requirements, and Challenges** presents four topics that include the problem of current technology-enhanced solutions for PBL, a basic idea proposed by our research group for driving PBL process design and implementation, the requirements for the technical solution, and the challenges to overcome these requirements.

5. **Applying Model-driven Approach** is a transitional chapter that presents how to effectively make use of a model-driven approach to transform the theoretical findings and ideas to a technology realizable solution.

6. **System Design** presents the concrete technical design of the system by illustrating a basic architecture and component structure detail and presents the solutions to solving the major problems, such as how to support PBL

authoring, how to design runtime logic, and how to manipulate PBL model data.

7. **Implementation** shows the final system by depicting four typical sets of user interfaces. These user interfaces involve a PBL authoring tool, a PBL course management tool, a PBL online whiteboard, and a PBL language editor.

8. **Usability and Acceptance Evaluation** presents a pilot study of investigating the usability and the acceptance of the modeling concept of the authoring tool provided by the designed system. This investigation received general positive feedback.

9. **Teaching Training Evaluation** presents a pilot study about whether the PBL authoring tool can be used to train teachers in designing online PBL courses. The evaluation showed positive responses.

10. **Summary and Outlook** concludes the work by providing a summary, listing the possible improvements for the system, and suggesting new orientations for future research and developmental work.

The content of this thesis is comprised based on the following scientific peer-reviewed articles. One was a journal article:

- Wang, D., Samaka, M., Miao, Y., Ali, Z., and Hoppe, H. U. (2016). *A Model-Driven PBL Application to Support the Authoring, Delivery, and Execution of PBL Processes.* Research and Practice in Technology Enhanced Learning, 11(6):1 (Wang et al., 2016).

Others were published in international proceedings:

- Wang, D., Miao, Y., Hoppe, U., and Samaka, M. (2014). *A Flexible System and Data Model for the Representation and Management of PBL Scripts.* In 22nd International Conference on Computers in Education. (Wang et al., 2014b).
- Wang, D., Miao, Y., Hoppe, U., and Samaka, M. (2014a). *A Domain-Specific Modeling Language Approach to Support Various Forms of Online PBL.* In 14th IEEE International Conference on Advanced Learning. IEEE. (Wang et al., 2014a).
- Miao, Y., Wang, D., Nongho, M. F.-P., and Samaka, M.(2015b). *Towards a Web-based Adaptive Problem-based Learning application.* In 15th IEEE

International Conference on Advanced Learning Technologies. IEEE. (Miao et al., 2015c).

- Miao, Y., Samaka, M., and Wang, D. (2015). *A Problem-oriented Approach to Represent and Manage Knowledge in PBL.* In 23rd International Conference on Computers in Education. (Miao et al., 2015a).
- Miao, Y., Samaka, M., and Wang, D. (2014a). *Plate Work-Bench: A PBL Authoring Tool.* In 11rd European ConFerence on Technology Enhanced Learning. (Miao et al., 2014).
- Miao, Y., Samaka, M., Wang, D., Ali, Z., and Romanowski, M. (2014). *Using a PBL Authoring Tool to Train Teachers in Designing an Online PBL Unit.* In 22nd International Conference on Computers in Education. (Miao et al., 2015b).
- Ali, Z., Wang, D., Samaka, M., and Miao, Y. (2016). *PLATE-PBL: Development and Implementation of a Script-based PBL Environment in Moodle.* In 16th IEEE International Conference on Advancing Learning Technologies. IEEE. (Ali et al., 2016).
- Samaka, M., Miao, Y., and Wang, D. (2016). *Support peer assessment processes in online problem-based learning.* In Global Engineering Education Conference (EDUCON), 2016 IEEE, pages 488-497. IEEE. (Samaka et al., 2016).

Chapter 2

Theoretical Background

2.1 Constructivist Approaches to Learning

Learning is a kind of cognitive process that is very complex. The extensive body of interpretations and theories imply this complexity. In order to deeply understand learning in the problem-based specific domain, it is necessary to understand the general principle of learning first.

From the constructivists' point of view, learning was a change in meaning constructed from experience (Newby et al., 2000) and thus learning was an active process of internally constructing knowledge rather than a passive process of directly memorizing external existing. The cognitivists had similar views, but from a different perspective that emphasized the acquisition of knowledge and internal mental structures, and claimed that learning was a mental activity, that knowledge acquisition was accomplished entailing internal coding and structuring by people themselves. When people learned, they were active participants in activities of knowledge acquisition (Bower et al., 1981). The world outside the human mind only provides information, and for cognition, humans need to process the given information to understand it. The activity of transforming information to knowledge called a learning process. The transformation is the core of learning. When people understand how to apply the knowledge in different contexts, we know that the transfer has occurred. In other words, external information is not knowledge until the information has the meaning created by people's own experiences and constructed as a new experience (Duffy and Jonassen, 1992; Ertmer and Newby, 2013).

Another imperative fact is that learning is engaged not only by the complex interplay among people's existing knowledge and collaboration activities between each other but also relying on the problem to be resolved.

In fact, during normal human development, there are always some problems to stimulate him or her to learn. The stimulus comes from random troubles in daily life, from questions asked by teachers, or because of certain goals, etc. As Savery and Duffy (1995) said: "cognitive conflict or puzzlement is the stimulus for learning and determines the organization and nature of what is learned." However, if learning only rely on the problems that randomly occurred in daily life, then learning will be very inefficient and fragmented, For this reason, people need man-made and well-designed problems. Bransford et al. (1977) mentioned that students were better able to construct new knowledge when they could relate it to what they already knew. That meant carefully structured learning activities were also needed. Summarizing these two points, to achieve efficient and systematic learning, the appropriate stimuli and guides must be provided Britain (2004). The activity of providing stimuli and guides is usually referred to as instructional design (or learning design, details will be given in Section 2.3). Ertmer and Newby (2013) said,

> "Instruction is structured around the presentation of the target stimulus and the provision of opportunities for the learner to practice making the proper response. To facilitate the linking of stimulus-response pairs, instruction frequently uses cues (to initially prompt the delivery of the response) and reinforcement (to strengthen correct responding in the presence of the target stimulus)"

where the cues corresponded to problems and reinforcement meant that interaction in between peers or learners and instructors.

We tend to call people who perform learning activities under instruction **learners**, and we call learners who are engaged in study activities in particular educational establishments, such as universities, schools, training institutions, etc. **students**. Based on the context of this research, I will generally use the term students rather than learners.

2.2 Problem-based Learning

From the previous section, effective and efficient learning was a learner-centered, self-directed, and often collaboration. It was an active cognitive behavior that turns external information into internal knowledge. Appropriately driven problems acted as stimuli to effective learning, and well-structured instructions guide the learner. Problem-based learning (PBL), as a pedagogy in which students learn about a subject through the experience of solving an open-ended problem, was developed in order to encourage students to learn in this type of active manner.

2.2.1 Background and Aims

Although there were numbers of different interpretations and theories in human learning, constructivism was considered the dominant educational theory. Within the last two decades, this theory was embraced by a lot of educational reforms that involve the generation of social constructivism, situated learning, and connectivism (Karagiorgi and Symeou, 2005). These constructivist theories later became the contemporary basis for the majority of current teaching-learning methods. From new developments to the general acceptance of constructivism, new theories conceptualized learning as both a matter of the person themselves and a matter of our society (Sharples et al., 2005). Kay (2010) pointed out that, generally speaking, these new theories additionally emphasized:

- the motivation of learners who desired and needed learning experiences that promoted high levels of interaction and activity;
- the development of technologies that allowed for immediate and effective access to information;
- and the demands of employers now expecting learners to acquire relevant 21st-century competencies such as critical thinking, problem-solving, creativity, etc. before entering the workforce.

As a constructivist teaching-learning method, Problem-based Learning[1] (PBL) emerged to be an important educational method that included the essences of effective and efficient learning from the traditional constructivism's perspective and the modern theory extensions mentioned above.

[1]Problem-based learning is very similar to inquiry learning (IL). There is no clear-cut distinguishing feature between them (Hmelo-Silver et al., 2007). To a large degree they reaches the same goal; they only come from different origins.

The principle idea behind problem-based learning was that the problem was a query or a puzzle that learners wish to solve (Boud, 1985). Later, Boud and Feletti (1997a) described problem-based learning more generally as "an approach to structuring the curriculum which involves confronting students with problems from practice which provides a stimulus for learning". Mayo et al. (1993) mentioned that problem-based learning posed significantly, contextualized, authentic situations, and provided students with resources, guidance, and instruction when students applied domain knowledge and developed problem-solving skills. With this learning approach, students were usually organized to work in small groups collaboratively to investigate and solve problems. By this way, they spontaneously found what they needed to learn (Barrows, 1996).

Contrary to the conventional teaching approach in discipline-centered curricula, Hmelo-Silver et al. (2007) pointed out that in PBL, teachers played the role of facilitating students during the learning process and provided content knowledge on a just-in-time basis rather than directly telling students what to know. Consequently, students cognitively engaged in sense making, developing evidence-based explanations and communicating their ideas. Several researchers said that this approach drove students to actively and collaboratively construct extensive and flexible knowledge bases and develop their critical thinking, problem-solving, teamwork, and self-directed learning skills through staged sequences of problems that were presented in an appropriate context, together with associated learning materials and support from facilitators or teachers (Barrows and Kelson, 1995; Boud and Feletti, 1997b; Hmelo-Silver and Eberbach, 2012). In fact, as common sense suggests and Section 2.1 presents, embedding learning in problem contexts is necessary to encourage students to develop flexible knowledge and effective problem-solving skills. This method and its success was demonstrated and supported by numerous research (Needham and Begg, 1991; Gallagher et al., 1992; Hmelo, 1998; Schwartz and Bransford, 1998; Hmelo et al., 2000; Hmelo-Silver, 2004).

2.2.2 Characteristics

According to Miao (2000); Hmelo-Silver (2004), PBL strategy is distinguished from other teaching approaches, especially due to the following characteristics:

- **Realistic ill-structured problems**. Real-world problems are often ill-structured with missing information and mixed knowledge from biology,

chemical, environmental science, etc. Jones et al. (1994) said, "Missing information will help them understand what is occurring and help them decide what actions, if any, are required for resolution." Students have to make decisions and provide solutions to the ill-structured problems and are driven to apply their prior experiences to identify inconsistent information, thus discovering their missing knowledge.

- **Problems that focus on information analyzing and reasoning strategies**. In PBL, students have to identify what learning issues they need to study and what information they should collect rather than simply being given all the issues and information. They are encouraged to collect information, find evidence, and use their prior knowledge to construct hypotheses and solutions in order to develop their reasoning skills. There is no right way or fixed formula for conducting the investigation, because each problem is unique and has no single "right" answer (Stepien and Gallagher, 1993).
- **Prompt self-directed learning processes**. Students take charge of learning, using self-directed learning in comparison students who learn the same content with traditional methods in traditional classrooms. Students individually need to have individual learning plans. Students define learning requirements based on goals and issues that are meaningful to them. They decompose the overall problem into sub-problems. They decide on action paths to resolve the sub-problems. They allocate resources, identify facts for each action, and generate their individual ideas, revisiting, and reflection by themselves.
- **Extensive collaboration**. Students are prompted to interact with other people, negotiate ideas, discuss problems with (naïve) understanding, debate different perspectives, and contribute their best efforts to each other in a social collaboration context. Under certain learning organizations, some information is gained through a collaborative investigation and collection. Sometimes they are required to go outside the classroom in diverse social contexts, such as at home, in laboratories, on streets, in factories, etc. to find learning information. Later the gathered information is shared within their learning groups or among different groups to construct shared knowledge. Jones et al. (1994) said, "they see themselves and ideas as others see them, articulate their ideas to others, and are fair-minded in dealing with contradictory or conflict views."
- **Changed role of students and teachers**. Unlike in traditional class-

rooms, students are not only the implementers and performers under teachers' "what to do" instructions, but they are also problem finders and solvers in anticipating, exploring, analyzing and solving problems. They also play the role of planners and producers, who are responsible for planning and designing methods to solve problems; become initiators and organizers to initiate tasks, coordinate work, or even facilitate others; become communicators and negotiators focused on expressing ideas, information, intentions, and feelings; or become others. The role of the teachers also changed from information givers to facilitators (or coaches and tutors). They are responsible for observing the learning process, providing rich environments, experiences, and activities; guiding students for performing reasoning and self-directed learning in right directions, giving feedback to help students build meaningful reflections; training them how to learn; and providing content knowledge on a just-in-time basis.

Boud and Feletti (1997b) described PBL in the following sequence style:

1. Using stimulus material to help students discuss an important problem, question or issue.
2. Presenting problems as a simulation of professional practices or a "real life" situations.
3. Appropriately guiding students' critical thinking and providing limited resources to help them learn from defining and attempting to resolve given problems.
4. Having students work cooperatively as a group, exploring information in and out of class, with access to tutors (not necessarily a subject specialist) who knows the problem well and can facilitate the group's learning process.
5. Getting students to identify their individual learning needs and appropriate use of available resources.
6. Reapplying this new knowledge to the original problem and evaluating their learning results.

2.2.3 Different Practical Models

According to Boud (1985), PBL was considered as one of the a most significant innovations in education (Boud, 1985). Many scholars applied this theory to educational practices. However, until now, there is no universally agreed upon practice to implementing PBL. Currently, PBL can be conducted in a number of ways based on different practical models or frameworks developed from different perspectives of emphases. These models or frameworks include the McMaster PBL model (Woods, 1996), the Maastricht "Seven jumps" framework (Gijselaers, 1995), Woods' model (Woods, 2000), Mills' five steps model (Mills, 2006), Seymour's "five-stage" model (Seymour, 2010), SALFORD model (McLoughlin and Darvill, 2007), etc. as well as some well-known practices published on the Internet such as IMS (2003); IMSA (2016); Gulfcoast (2016), etc. For instance, among in these models or frameworks, a relatively popular one is the Maastricht "Seven jumps" framework. It consists of the following seven steps(Gijselaers, 1995):

Step 1: Clarify terms and concepts not readily comprehensible.
Step 2: Define the problem.
Step 3: Analyze the problem.
Step 4: Draw up an inventory of the explanations discussed in step 3.
Step 5: Formulate learning objectives.
Step 6: Collect additional information outside the group.
Step 7: Synthesize and test the newly acquired information.

IMS (2003) noted that there is a similar PBL tutorial practice or variant on the Internet. It is described as the following:

Step 1: Identify and clarify unfamiliar terms presented in the scenario; list those that remain unexplained after discussion.
Step 2: Define the problem or problems to be discussed; students may have different views on the issues, but all should be considered; record a list of agreed upon problems.
Step 3: "Host brainstorming" session to discuss the problem(s), suggesting possible explanations from prior knowledge; students will draw on each other's knowledge and identify areas of incomplete knowledge; record full discussion.
Step 4: Review Steps 2 and 3 and arrange explanations into tentative solutions; scribe possible organizations, explanations, and restructures if neces-

sary.

Step 5: Formulate learning objectives; groups will reach consensus on learning objectives; tutors will ensure learning objectives are focused, achievable, comprehensive, and appropriate.

Step 6: Private study; all students gather information related to each learning objective.

Step 7: Group shares results of private study; students will identify their learning resources and share their results; tutor will check learning and may assess the group.

Hmelo-Silver (2004) claimed that the PBL could be abstracted as a learning cycle where

> the students are presented with a problem scenario. They formulate and analyze the problem by identifying the relevant facts from the scenario. This fact-identification step helps students represent the problem. As students understand the problem better, they generate hypotheses about possible solutions. An important part of this cycle is identifying knowledge deficiencies relative to the problem. These knowledge deficiencies become what are known as the learning issues that students research during their self-directed learning. Following self-directed learning, students apply their new knowledge and evaluate their hypotheses in light of what they have learned. At the completion of each problem, students reflect on the abstract knowledge gained. The teacher helps students learn the cognitive skills needed for problem solving and collaboration. Because students are self-directed, managing their learning goals and strategies to solve PBL's ill-structured problems (those without a single correct solution), they also acquire the skills needed for lifelong learning.

Figure 2.1 shows the learning cycle above. Figure 2.2 illustrates PBL from another angle that focused on the constituent elements. Gijselaers (1995) considered that PBL consists of three kinds of input variables: (1) student characteristics which include students' prior knowledge about the subject matter of a course, (2) features of the problems used in a course, and (3) tutor's teaching skills. The learning activities are performed in small groups under appropriate additional instructions. The major task is the self-directed learning, which is as Hmelo-Silver (2004) said. The outcome is learning results and further motivation.

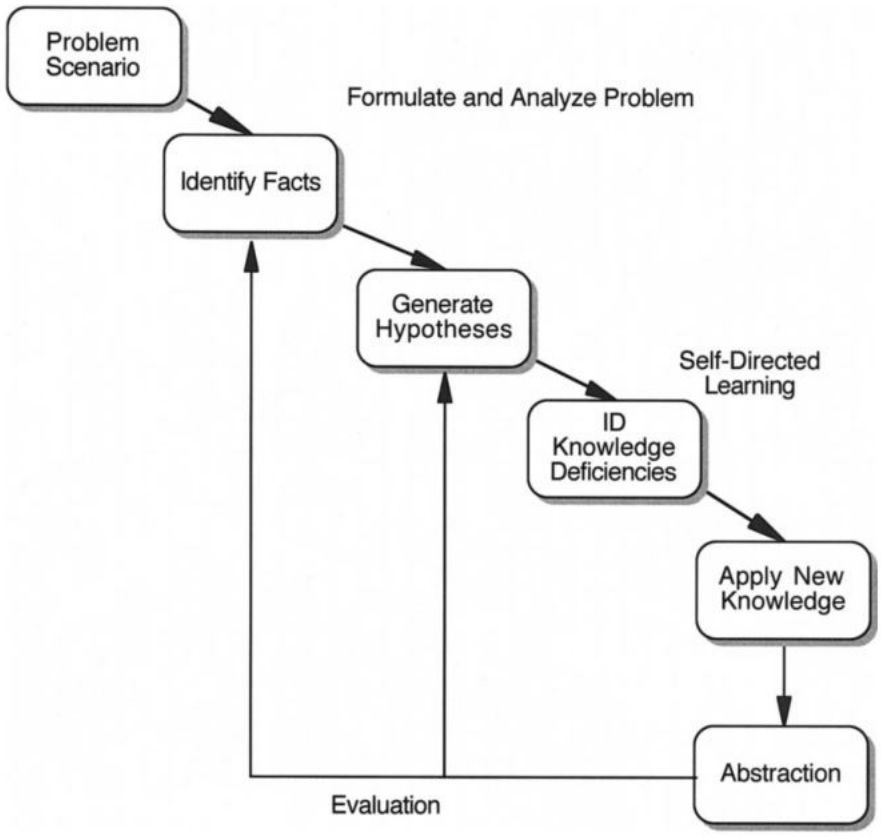


Figure 2.1: The problem-based learning cycle (Hmelo-Silver, 2004).

2.3 Learning Design

Dalziel et al. (2003) said, "Learning design has emerged as one of the most significant recent developments in e-learning[2]", and it is also the core concept of technology enhanced PBL.

2.3.1 Concept of Learning Design

Broadly speaking, learning design is far from being a new concept. In the context of face-to-face teaching in traditional classrooms, teachers consciously made their daily lesson plans based on their general sense, although they could have paid too much attention to noticing or realizing the model design of learning processes. That is, they usually subconsciously develop learning activities, which is currently

[2]With the continuous progress of technology in the industry, the substance of e-learning is also changing all the time. Here in this thesis, the author considers that e-learning nowadays indicates the information and communications technology (ICT) enhanced learning or simply technology enhanced learning.

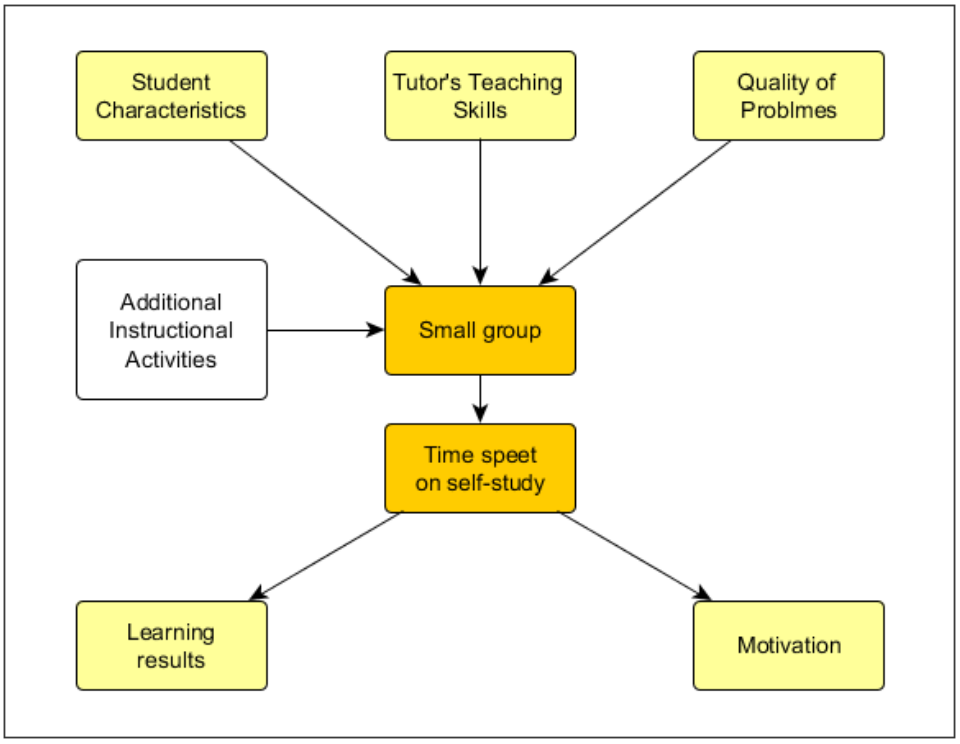


Figure 2.2: The abstract constituent elements of PBL (Gijselaers, 1995).

seen as learning design, when there is a need in teaching sessions (Britain, 2004). In the "Handbook on information technologies for education and training" (Adelsberger et al., 2013), R. Koper and S. Bennett described the learning design as the following:

> The primary role of any instructional agent, whether it be a teacher, the learners themselves or a computer, is to stimulate the performance of learning activities that will gradually result in the attainment of the learning objectives. The instructional agent defines the tasks, provides the contexts and resources to perform the tasks, supports the learner during performance of the task and provides feedback about the results. The learning activities that are needed to obtain some learning objectives are, in most cases, carefully sequenced according to some pedagogical principles. This sequence of learning activities that learners undertake to attain some learning objectives, including the resources and support mechanisms required to help learners to complete these activities, is called a **learning design**.

However, in the context of e-learning, where learning is conducted via electronic media, e.g. using computers to deliver part or all of a course, the concept of learning design indicated the way that teachers use a formal description to structure the sequence of learning activities to reflect the learning needs of students. In this way the structured sequences of information and activities can be carried out via computers, together with the necessary supports from the teacher, to promote technology enhanced learning (Britain, 2004; Bennett et al., 2005; Conole and Oliver, 2006). In this thesis, we talk about learning design in this context. For the reason of making the description in this thesis clearer, in all the following text, I will use the following convention: When **learning design** is noted with a lowercase **l** and **d**, it indicates the general concept stated by Adelsberger et al. (2013); and when **Learning Design** is noted with capital **L** and **D**, it indicates the concept in the context of technology-enhanced learning. This paragraph will further talk about **Learning Design**.

Educational practitioners widely appreciated the Learning Design. For instance, CETIS and several others made many workshops on Learning Design, and the results showed the level of interest in Learning Design among practitioners and others was both extensive and deep. Therefore it was named as the preferred strategy for the Department for Education and Skills for eLearning United Kingdom (Olivier, 2006). Besides, researchers also considered that the formal Learning Design is useful for improving education (Bennett et al., 2004). Especially, it was proved that the teachers could readily understand a textual and graphical form of Learning Design and that Learning Design's re-usability was a very practical mechanism to guide teachers in designing problem-based learning processes for students (Bennett et al., 2005).

As a continuously growing distinct field, the goal and the concept scope of Learning Design was extended. This growing extension showed in two aspects. First, in the beginning, Learning Design focused on designing learning content and assessment of outcomes for knowledge transmission from the perspective of teachers. Whereas modern Learning Design focused on planning, structuring, and sequencing learning activities and designing learning context and environment with technical support for knowledge construction from the perspective of learners. Examples of such learning activities include: identifying and analyzing problems in a session, brainstorming learning issues using a digitalized whiteboard, gathering information from the Internet using a search engine, proposing and discussing solutions in a discussion forum, and co-authoring group reports using a Wiki, etc.

The other aspect, in the beginning, is that Learning Design aimed at providing a means to represent and communicate the designs of learning activities so that they could be shared among practitioners. Later, the term Learning Design also denoted the result or product of the design process—a computational description of the teaching-learning process that occurred in lessons or courses with the help of computers, which served as the means to orchestrate and scaffold teaching and learning practice at run-time (Koper and Tattersall, 2005; Miao et al., 2005).

As we have mentioned, the designing of a course plan is a creative, complex and iterative task. It tended to be implicit, not formally articulated, or externalized for others. Teachers heavily rely on prior knowledge and experience in their design practice and rarely follow any formal design method processes (Conole et al., 2007). In particular, creating a high quality, technology-supported learning process was a significant challenge for educators (Lockyer et al., 2009). Regarding research and assistant system development, supporting Learning Design is usually concerned with the development of methods, tools, and resources for helping teachers in their teaching-learning practice(Koper, 2001; Lockyer et al., 2009; Beetham and Sharpe, 2013).

2.3.2 General Ideas Behind

Unlike traditional face-to-face learning, which saw learning objects as the core entity, nowadays Learning Design focuses on learning activities, content, and servicing learning (inter)actions. Britain (2004) presented the following general idea behind Learning Design:

- People learn better when they are actively engaged in learning activities.
- Sometimes learners cannot learn effectively and efficiently by themselves. These learners need the benefit from the guidance and support in their learning activities.
- Learning activities must be designed carefully and deliberately to be a sequenced or structured learning workflow to accomplish more successful teaching.
- Successful teaching involves different strategies and techniques to engage, motivate and energize students. However, many teachers teach it without well-designed learning guidance and support.
- A Learning Design framework can push teachers to reflect in a deeper and

more creative way when they design and structure activities for learners.
- Learning Design is especially helpful for recording learning workflow and its contents for the purpose of communication, sharing, and re-use in the future.

From the perspective of the composition, generally speaking, Learning Design consists of:

- **Information objects**—Structured aggregations of digital assets, designed purely to present information, where the asset typically indicates a single file (e.g. an image, video or audio clip), sometimes called a 'raw media asset' (Conole and Oliver, 2006).
- **Learning activities**—Tasks involving interactions with the *Information objects* to attain a specific learning outcome (Conole and Oliver, 2006).
- **Structure**—Sequence and relationship of *learning activities*.

Information objects were also called 'information content,' which did not have learning or teaching effects unless they are placed within learning activities in Learning Design. Under Learning Design concepts, there were a very important instantiation, called IMS-LD (IMS Learning Design Specification[3]). The most significant influence of IMS-LD was that it made possible (or at least aiming at providing the possibility) describing almost all types of teaching-learning scenarios. Because it consisted of pedagogy-generic terms such as roles, activities, properties, and conditions abstracted from more than 100 pedagogical approaches (Koper and Manderveld, 2004; Koper and Tattersall, 2005), which included problem-based learning, it is able to describe problem-based learning processes.

2.3.3 IMS Learning Design

Although the work presented in this thesis did not heavily rely on IMS-LD specification, it was still very necessary to understand the IMS-LD first. This was because (1) the thinking underneath the IMS-LD was a significant theoretical reference for my work, (2) as mentioned in Chapter 1, our system should have interoperability with IMS-LD compatible tools, applications or systems.

IMS-LD was a standardized Learning Design language. It was developed by Koper and Olivier basing on the Educational Modeling Language (EML) (EML, 2002). The language was a notational system aiming at providing an abstract way to

[3]The full name of IMS-LD is Instructional Management System Learning Design Specification.

describe teaching and learning interactions as a process pattern. IMS-LD was designed with the consideration of (1) integrating existing specifications as much as possible, and (2) representing different pedagogical models (Koper, 2001) of lessons or course plans, or best practices in unit-of-learnings (UoL[4]) in a formal way in order to generally provided a possibility of shaping the design of teaching and learning in the e-learning context. The first version of the specification was published (IMS, 2003). The complete idea and details could be found in "Representing the learning design of units of learning" by Koper and Olivier (2004).

As a result, according to Olivier (2006), the specification was designed to (1) provide the capability of describing and implementing different kinds of learning approaches; (2) enable repeatable, effective, and efficient *units of learning*; (3) provide access to and interchange the *units of learning* between learning systems; (4) support multiple delivery models; (5) support the reuse and re-purposing of *units of learning* or their component elements; (6) leverage existing specifications and standards; (7) be culturally inclusive and accessible (internationalization); and (8) support reporting and performance analysis. Those features was similar to the aims of the PBL scripting language mentioned in the PBL project.

As a notation standard, IMS-LD was usually referred to as a Learning Design language. In fact, it belonged to a type of modeling language. In this section, I try to provide an understanding of this language from another angle. Koper and Olivier (2004) provided a big picture showing the core concepts of IMS-LD, which are detailed in Figure 2.3. These concepts constituted a meta-language to generically describe learning activities. The blocks indicated the type system of this modeling language, while the relations indicated the syntax for the modeling.

From an entity–relationship point of view, most of these blocks could be divided into two categories. The first could be called constructional entities which include *person*, *activity*, *environment*, *play*, *act*, etc. These blocks were easy to understand because they weren't complicated or new concepts. They were simply the participating elements in the teaching-learning process. The second category could be called representational entities including *method*, *role*, *activity-structure*, etc. For instance, *method* or *activity-structure* presented how some or all of the participat-

[4]UoL is a digital package for representing one or more learning objectives. It can describe a course, a module, a lesson, or a single activity (e.g. a discussion) and includes references to the digital and non-digital learning objects and services needed during learning. It is in contract to the content packaging standards, such as QTI (QTI, 2005) and SCORM (ADLI et al., 2001), which are content-oriented. And the user engaged in the output model based on those standards is just a single learner or student.

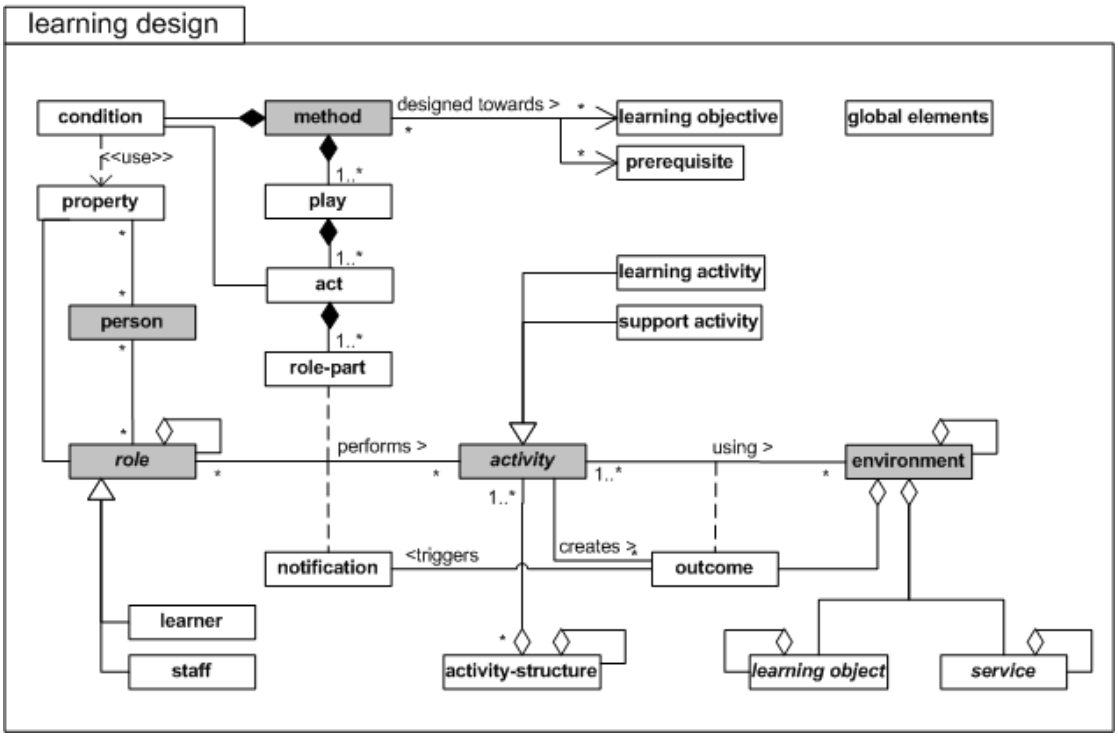


Figure 2.3: The conceptual structure of the IMS-LD specification (Koper and Olivier, 2004).

ing elements organized and worked together. *Role* was a label name that presented how a person (a participant of a teaching-learning process) was involved in an activity in a specific period in a specific context. These representational entities were relatively special and could be seen as the core capability in expressiveness in this language. Exactly as Adelsberger et al. (2013) said, "The method section is the core part of the Learning Design specification in which the teaching-learning process is specified. All the other concepts are referenced, directly or indirectly, from the method." Of course, there were also some other entities. An important one of them was *condition*.

I designed an additional diagram to further explain how the IMS-LD specification works. In Figure 2.4, the *roles* labeled *persons* who could be learners or teaching staff. The name of the *role*, however, was not important. *Roles* were allocated to activities. Since different *persons* could play different roles, different activities were allocated in different environments. Consequently, these provided a semantic express that different persons performed different activities in different environments, which constituted different teaching-learning sub-scenarios. But this expression was still not enough because these scenarios were not organized in a

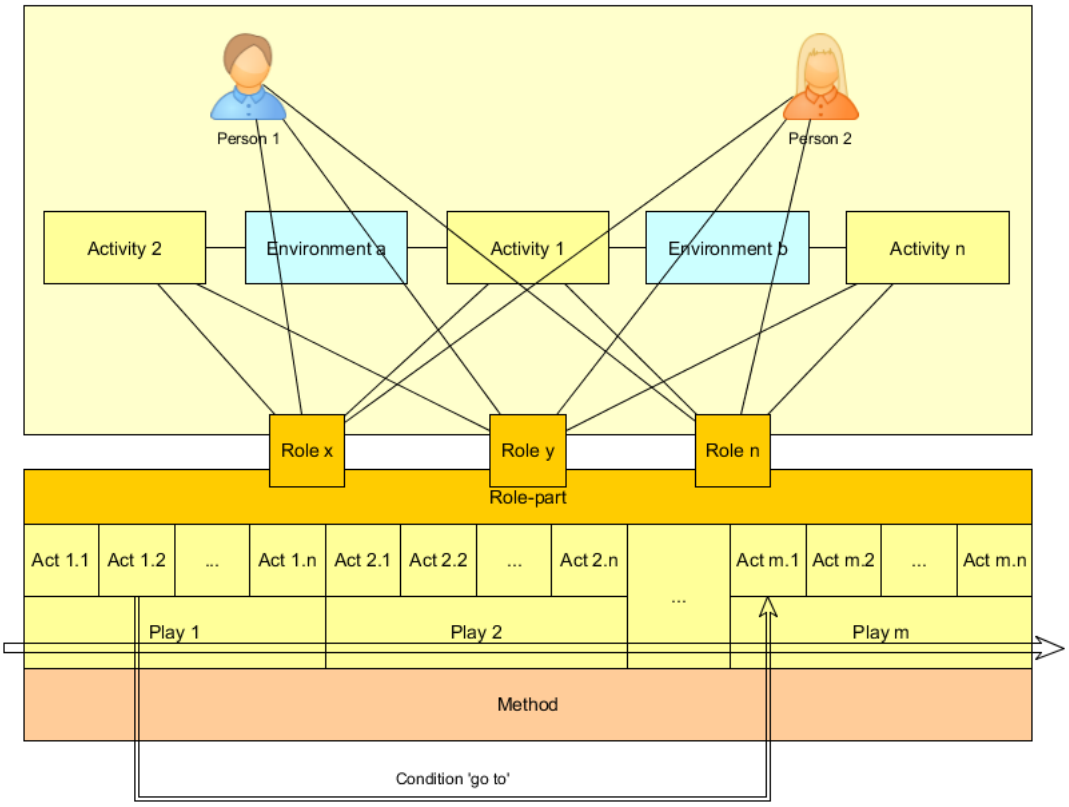


Figure 2.4: A view of how the IMS-LD specification works.

certain order. However, the entity *method* provided this function. In consideration of providing a more fine-grained representation, the *method* was associated with representational entities (*play* and *act*) by using a metaphor of a theatrical play. Finally, *role-part*, which assigned an activity to a role, bound teaching-learning sub-scenarios and organization together to achieve the basic capability of expressiveness. From Figure 2.3, we could see that the activities and roles were reusable for different plays.

The expressiveness of the IMS-LD specification provided above was categorized to a class, so called Level A. Nevertheless, the expressiveness of Level A was still limited in comparison to some complex learning scenarios. For example, there existed the question of how to meet the requirements of personalized learning or adaptive learning. Thus there was also a Level B and a Level C in IMS-LD. Level B added *property* and *condition* to Level A. *Property* enabled the information about learners, roles and the state of Learning Design itself to be maintained. According to Miao (2005), this way was important for recording outcomes of participants and supporting personalization. In addition, Koper and Olivier (2004) said the *condi-*

tion provided "*If-Then-Else* rules that further refine the assignment of activities and environment entities for persons and roles, enable the 'go to' or 'skip' jump in a learning process, which may be used to the personalization for specific users." Level C added *notification* upon Level B. *Notification* provided an event-driven messaging system that enabled a better interactivity and control possibility of a learning process during the runtime. As Koper and Olivier (2004) also said, "This mechanism can be used to model adaptive task setting Learning Design, where the supply of a consequent activity may be dependent on the kind of outcome of previous activities."

However, IMS-LD was used not only for describing teaching-learning processes but also for handling the corresponding learning content and learning services: for example, third-party services such as online whiteboard. These could be coordinated with activities, teachers, and students through role-play settings, role-play sequence settings, different locations, and appropriate permissions. Learning tasks and resources would be populated properly in time to multiple or single learners and teachers by the IMS-LD runtime engine[5].

When we talk about IMS-LD as a language, it implies that we stand on the computer modeling point of view, which means this standard was designed not primarily for teachers to directly use, but for computer-supported learning. For example, according to the specification, a design outcome must be an XML format document. Writing this kind of file by hand was obvious a time-consuming and error-prone task. Thus design and implementation were recommended to be done through a software's help. In other words, to make design work easy to handle and run learning processes on computers, IMS-LD based authoring tools and corresponding run-time learning tools (runtime engines) were needed.

[5]A runtime engine is a machine agent who is responsible for making a computational description of a teaching-learning process, namely a Learning Design outcome, happen in a lesson or a course serving as a means to orchestrate and scaffold teaching and learning practices.

Chapter 3

Existing Tools

3.1 Overview

Information and Communication Technologies (ICT) had been applied to more and more aspects in eduction. This was due to the technology enhanced learning (also be referred to as e-learning mentioned above or computer supported learning in other literatures, etc.) providing a great number of benefits in comparison to the "old fashion" way. The advantages included scalability, cost-effectiveness, broad geographic reach, unmatched delivery speed, flexible to learn at any time in any place, etc. As a way of education, PBL also shared these technical reinforcement benefits.

All of the aforementioned benefits were actually a projection of the general advantages of the ICT in the area of the education. However, the education had its particular characteristics. Therefore, it was naturally to apply some special technologies to fit the special characteristics, in order to made the potential advantages of ICT be placed in the education domain as much as possible.

Specifically, to gain benefits from utilizing ICT for PBL, generally speaking, there were two ways, the one was using purpose-generic information and communication tools. In this way, teachers and students could combine use file management and sharing tools, such as interconnected PC, Dropbox, Google Drive, etc.; communication tools, such as mobile devices, Skype, etc.; or other tools like FreeStyler[1],

[1]FreeStyler is a cooperative and interactive modeling tool which combines different graphical languages being used to structure the content with hand written input. Web site: http://projects.collide.info/projects/freestyler-release/wiki/Wiki

Whiteboard, etc. However, these tools were not designed for systematical integration. Thus, teachers needed to organize the involved tools properly themselves. Since this way lacked the systematic automation for learning processes, the difficulties of organization, monitor, and management of learning activities were not reduced. Even worse, sometimes it increased difficulty. In fact, there was no real innovation in this way in terms of supporting PBL. Therefore, the state of art review in this thesis will not focus on this way.

The other way was to use the tools which designed to support and manage technology-enhanced teaching and learning content or activities. The usage of those tools for PBL could be further divided into three categories; they were (1) using content-oriented teaching-learning tools, (2) using activity-oriented teaching-learning tools, and (3) using blended teaching-learning tools. In the following sections, a detailed review will be presented under this classification.

3.2 Content-oriented Tools

The concept of learning content-orientation had a very long existing history. Currently, most of the learning management systems (LMSs) were developed based on this concept. Moreover, to a great extent, content-oriented LMSs had dominated technology enhanced learning. Since most LMSs were education genetic, a large number of LMSs could be adopted to assist PBL.

According to Ellis (2009), a learning management system was a software application for the administration, documentation, tracking, reporting, and delivery of electronic educational technology courses or training programs. LMSs were designed to deliver and manage instructional content, identify and assess individual and organizational learning or training goals, track progress towards meeting goals, and collect and present data for supervising the learning process of the organization as a whole (Szabo and Flesher, 2002). Third-party tools for particular purposes could also be developed based on this kind of system.

Nowadays, most learning content-oriented LMSs were very sophisticated due to the continuous improvement driven by the continuously emerged teaching-learning needs. Most LMSs were also very powerful due to the non-stop applying the state-of-art ICTs and functional improvement, e.g. most LMSs are Web-based and real-time cooperation supported now. Speaking in detail:

- Some LMSs were highly advanced due to providing the functionality of monitoring learning activities to improve the quality of teaching and learning. Typical monitoring included logging duration and frequency of learning units visited by learners, assessing results, computing statistics of learning behavior, etc.
- With unstoppable, intensive improvement, most popular LMSs became considerably powerful. This is because they are always using or trying to use the state of art ICTs for the purpose of either providing better performance (including low latency, high-speed transportation of data stream, and large volume of storage), achieving an overall capability of cross-platform (ranged from including different operating systems and Web browsers to various types of mobile devices), or supporting complex, massive, and real-time collaboration, etc.

Currently, the top level of learning management systems was applicable for commercial only. Of course, there were some great but non-profile LMSs. One of them was Moodle, which was quite popular and heavily used in schools, universities and other organizations.

Moodle was an open-source, Web-based, pluggable LMS platform. Its most important feature was that people could customize it to create websites with online courses for educators and trainers according to their specific needs. For instance, it could be extended to a PBL-specific learning system through adding an add-on. Section 3.4.2 will discuss this add-on with details.

3.3 Activity-oriented Tools

As mentioned in Section 3.1, there were three kinds of LMSs. The last section was about the content-oriented LAMs. In contrast, this section discusses the activity-oriented LMSs. Although these systems were just a small part of LMSs, the concept underneath these systems showed an important direction of technology enhanced learning (or problem-based learning). Activity-oriented tools could be also seen as learning design centered.

3.3.1 LAMS

Learning Activity Management System (LAMS) was a general-purpose learning design tool, created by James Dalziel of Macquarie University in Sydney and WebMCQ Ltd (Dalziel et al., 2003). LAMS embodied the core ideas behind Learning Design regarding a focus on creating sequences of activities, which was inspired by EML and IMS-LD. Rather than emphasizing content, it provided teachers a simple but highly intuitive graphical authoring interface for creating sequences of learning activities, including individual tasks, small group work, and whole class activities. Its user-friendly user interface allowed teachers to drag and drop LAMS activities into its workspace and use directed connections to organize activities as a workflow. Learning content and collaboration configurations could be set to the activities. LAMS was now an open source learning design system. It supported exporting IMS-LD Level A formats to assist teaching-learning activities with interoperability in Version 2.2. Figure 3.1 shows a general user interface of LAMS.

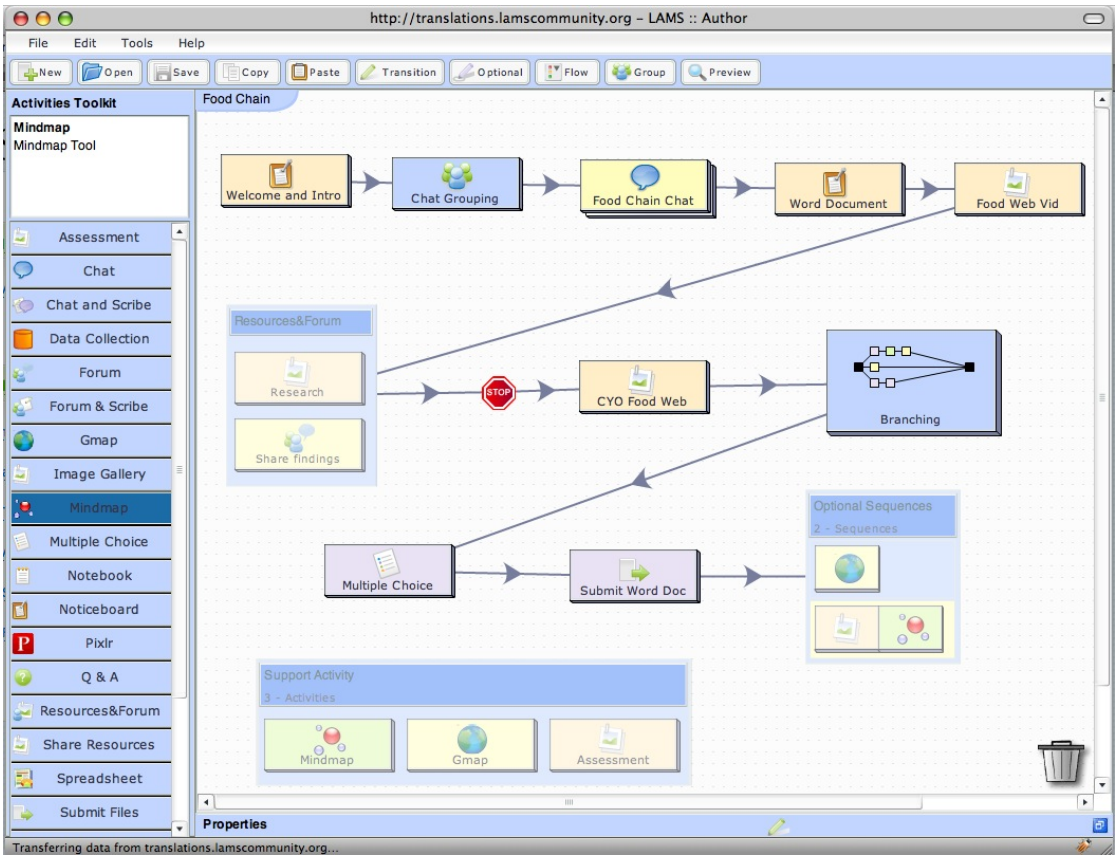


Figure 3.1: LAMS user interface for learning design.

3.3.2 CopperCore and CopperAuthor

CopperCore (OUNL, 2008) was developed by the Open University in the Netherlands (OUNL) as an open-source runtime engine for ISM-LD application. It was the first IMS-LD run-time engine that could handle learning process based on the description in *units of learning* and was also the first engine that supported all three levels of IMS-LD (Level A, B, and C). The latest version was 3.3, released in November 2008. However, CopperCore was not meant for teachers or learning designers to use, but for IMS-LD based software developers as an implementation reference. In fact, providing other software developers a definitive interpretation of the behavioral features of the IMS-LD specification was the primary significance of this tool.

CopperCore was also designed as a service to incorporate IMS-LD in other applications. It provided SOAP interfaces under J2EE technology as application program interfaces (API) covering the publication, administration, and delivery of IMS-LD based course models.

Although CopperCore was a runtime engine, it did provide a few user interfaces, mainly for the purpose of giving developers an intuitive about how a course moved from an idea to a computer supported runnable course. These user interfaces included a command line interface handling most functions, a publication interface for the example purpose, and a Web delivery interface also for the example purpose.

The Open University in the Netherlands also additionally developed a treetable plus form-based graphical IMS-LD authoring tool called CopperAuthor (OUNL, 2006). Figure 3.2 shows its typical interface. In comparison to the command line interface in CopperCore, CopperAuthor made it possible for teachers to use the IMS-LD notation model, validate their course plans, and produce *units of learning*. It was fully compliant with IMS-LD, supported Level A, and worked well with CopperCore. Its user target is teachers, specifically. It was also an open-source software.

3.3.3 RELOAD LD Tools, OpenGLM and MoCoLaDe

RELOAD was a project founded by the JISC Exchange for Learning Programme (X4L). Tools under this project that could be combined and used to support PBL. In the context of this chapter, RELOAD provided two relevant tools: a

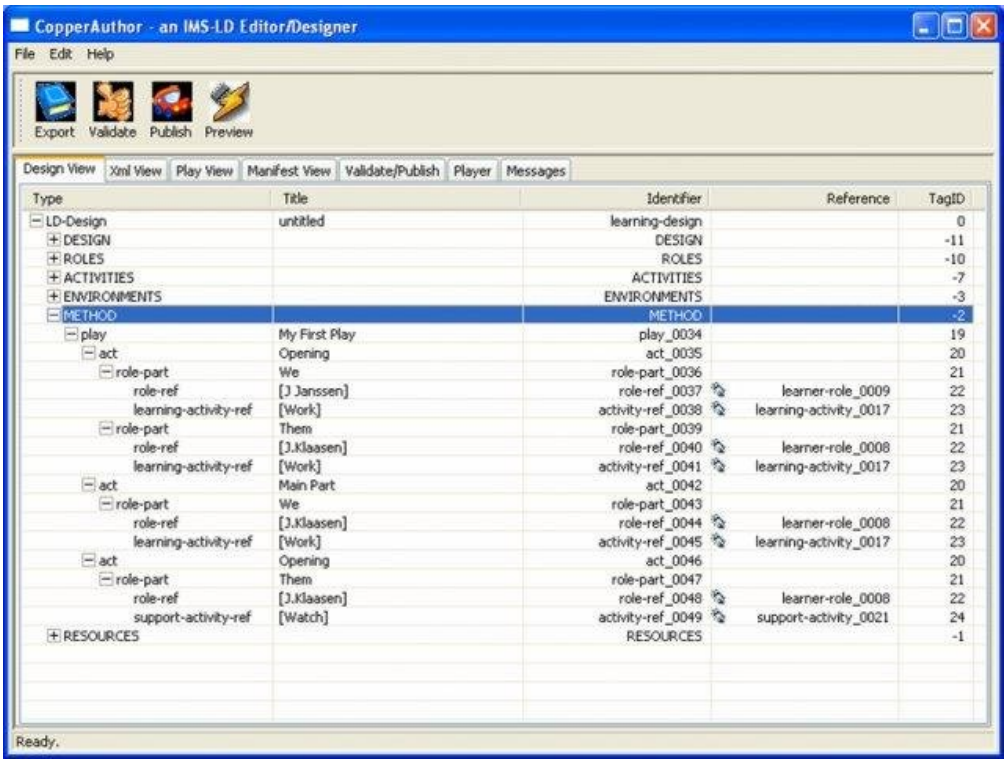


Figure 3.2: The user interface of CopperAuthor for IMS-LD.

Learning Design Editor (later was superceded by the new ReCourse LD Editor (TENCompetence, 2010)), and a Learning Design Player.

RELOAD Learning Design Editor was built on the Eclipse Rich Client Platform[2]. Therefore, it was a Java based application, which means it was cross-platform runnable. RELOAD Learning Design Editor (JISC, 2006; Griffiths et al., 2007) was designed to create IMS-LD compliant *units of learning*. It supported the full IMS-LD for Level A, B, and C, using a graphical user interface for the manipulation of all elements. This learning design editor also provided three fully functional LMS like components: a *project manager* for *Learning Design* organization, a *resource manager* for favorite files and web links organization, and a *file manager*. Figure 3.3 shows a screenshot of its *environments* page.

The RELOAD Learning Design Player was developed by embedding the CopperCore engine. It provided an easy to use management interface. It was also developed on the Eclipse Rich Client Platform, meaning it was also a cross-platform

[2]While the Eclipse platform was designed to serve as an open tools platform, it was architected so that its components could be used to build just about any client application. The minimal set of plug-ins needed to build a rich client application was collectively known as the Rich Client Platform.

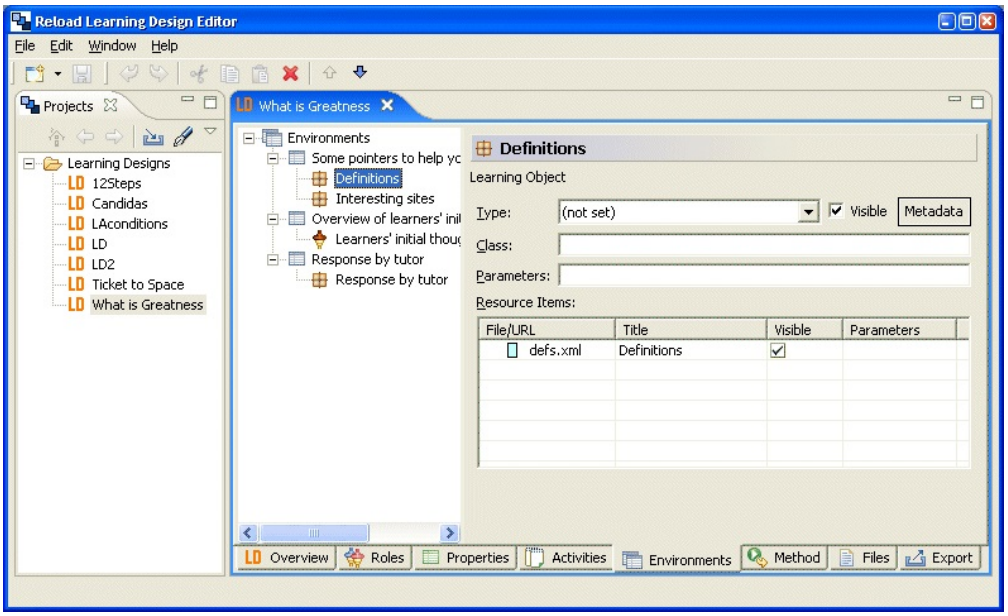


Figure 3.3: A screenshot of RELOAD Learning Design Editor's *environments* page.

software. Figure 3.4 shows a screenshot of the Learning Design Player in action.

OpenGLM (Open Graphical Learning Modeler) was developed under the ICOPER project[3]. OpenGLM was developed based on the code of the RELOAD Learning Design Editor. Therefore it was also a cross-platform softwarer. OpenGLM supported IMS-LD levels A and B.

In comparison to the RELOAD Learning Design Editor, OpenGLM's major enhancements included two aspects: (1) it provided a comprehensive and intuitive graphical user interface for IMS-LD modelling, in order to reduce the difficulty of building IMS-LD units of learning in teaching practice (Neumann and Oberhuemer, 2009). Figure 3.5 shows the intuitive graphical user interface; (2) it supported communities of practice in sharing IMS-LD units of learning along with standardised learning outcome definitions by providing built-in features for search, import from and export to the Open ICOPER Content Space (OICS[4]) (Derntl et al., 2011).

Additionally, there was a integrated learning design environment, shortly ILDE (Hernández-Leo et al., 2014), integrated OpenGLM as the Learning Design authoring tool and a virtual learning environment, so called CLUE!PS system, as

[3]http://www.ld-grid.org/resources/projects/icoper

[4]The OICS is a large openly accessible repository for different types of educational resources. The contributors the resources were worldwide, which includes OU's OpenLearn, OER Commons, MIT OpenCourseWare, etc.

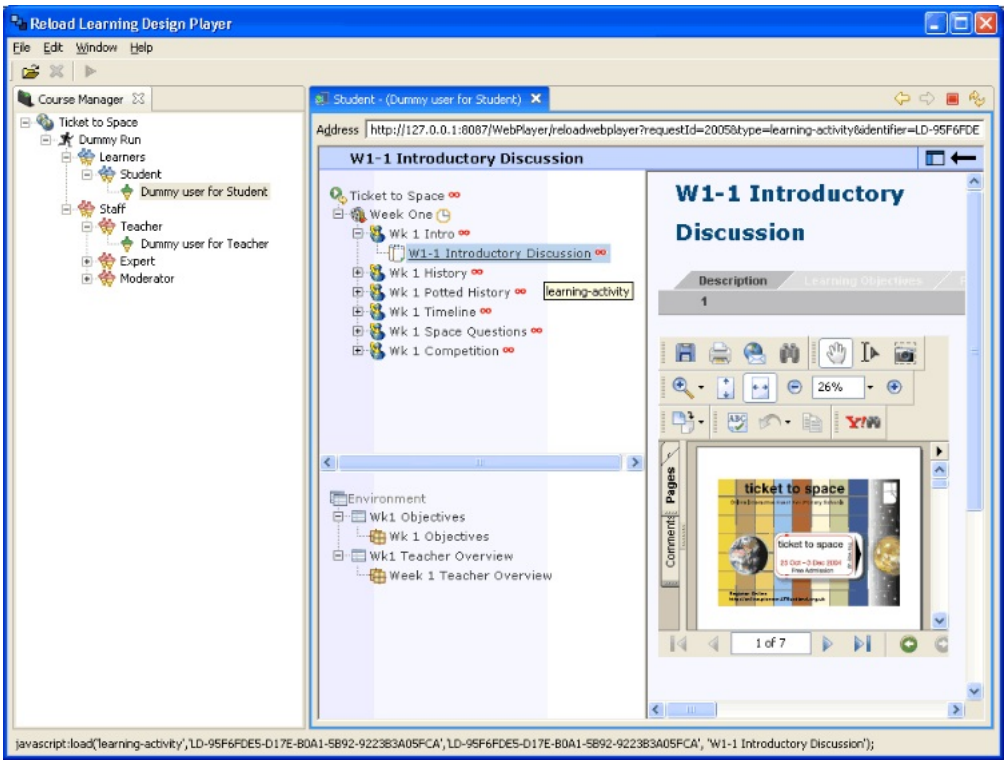


Figure 3.4: A screenshot of the RELOAD learning design player in action.

the runtime environment. Therefore, the Learning Design results of OpenGLM could not only be implemented in IMS-LD compliant runtime environments but also in the CLUE!PS system (Prieto et al., 2013).

Mocolade (Model for Collaborative Learning Activity Design) (Harrer et al., 2007) was a Freestyler (Lingnau et al., 2003) extension that also provided a easy to use graphical modeling user interface like OpenGLM and was able to export the learning process as a IMS-LD. Therefore the Learning Design results could be executed in the RELOAD Learning Design Player. Additionally, Mocolade could simulation the leaning process before it would really run, so that it enabled a practical validation function to check if the learning resources and roles were correctly distributed among users. Figure 3.6 shows the user interface of Mocolade.

3.3.4 MOT+LD

MOT+LD was an extension of MOT+, which was an object-oriented modeling editor (Paquette et al., 2005, 2006). The extension supported learning design helping people to identify and complete teaching-learning patterns in a diagram

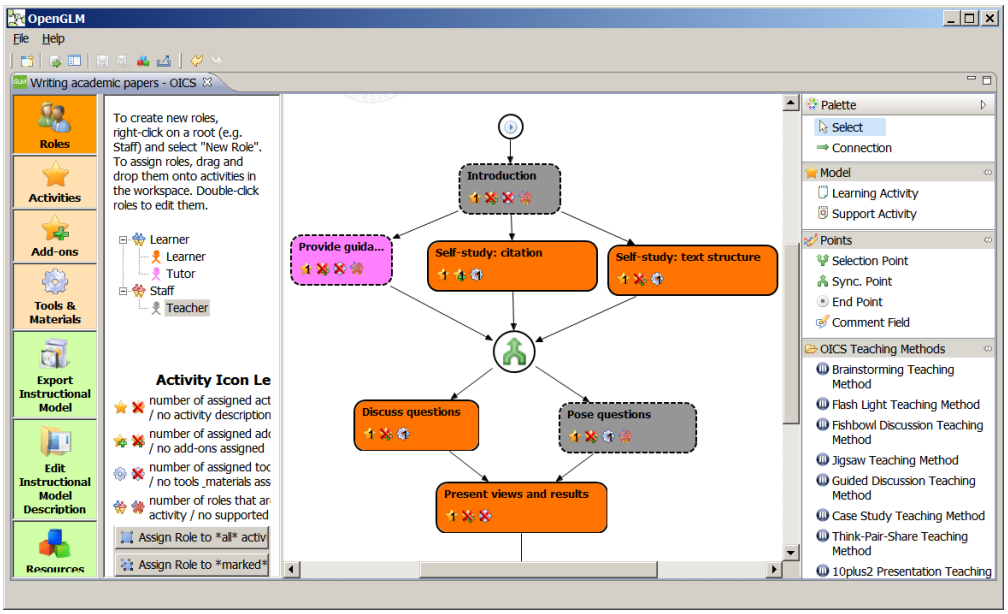


Figure 3.5: The main window of the OpenGLM (Open Graphical Learning Modeler).

manner. MOT+LD editor could be configured to create learning designs conforming to IMS-LD level A. Figure 3.7 shows this kind of mapping. It output the modeling results as *units of learning* to make IMS-LD run-time tools executable. This tool was designed for experts rather than classroom teachers to author learning process, although it did provide graphical representations that facilitated the authoring task to a certain extent.

3.3.5 ASK-LDT

ASK Learning Designer Toolkit (ASK-LDT) was another learning design tool following the IMS-LD specification for facilitating authoring teaching-learning plans (Karampiperis and Sampson, 2005).

Similar to ReCourse as introduced above, ASK-LDT also provided a graphical user interface, although in another style, for authoring the sequence of learning activities. The major difference was that ASK-LDT used a standard low-level notation language for the description of learning scenarios to be able to exchange learning designs between other systems that didn't follow IMS-LD (see Figure 3.8). It also enabled pedagogical designers to use their design notation, or high-level notation, to define learning scenarios according to their particular requirements(see Figure 3.9).

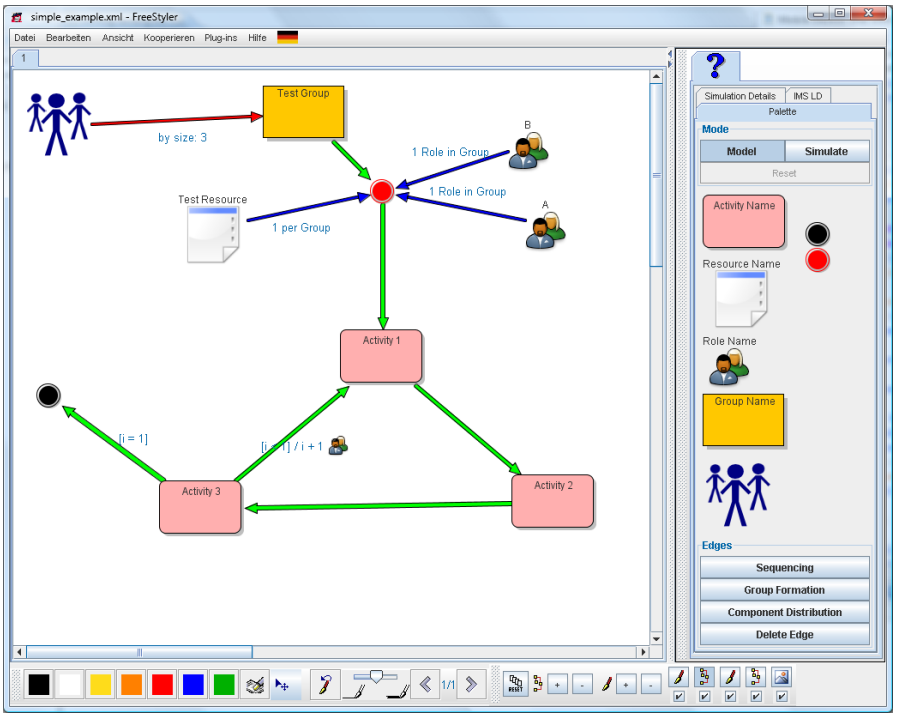


Figure 3.6: The Learning Design user interface of Mocolade (Model for Collaborative Learning Activity Design).

An upgraded version (Version 2.0), a Web-based ASK-LDT was developed and supported IMS-LD up to Level B.

3.3.6 Other Systems

There were some other tools, such as COSMOS (Miao, 2005) (Figure 3.10), eLive LD-Suite (Olivier, 2006) (Figure 3.11), SLED (McAndrew et al., 2005), Edubox (Tabbers et al., 2004), etc. that were either undergoing maintenance or closed. Although they may have had good design ideas, e.g. COSMOS, or had good user interfaces, or other advanced features, considering that they were not available for use anymore and for the reason of saving space, they will not be featured in this thesis more than using screenshots for the purpose of clarifications for other projects or ideas.

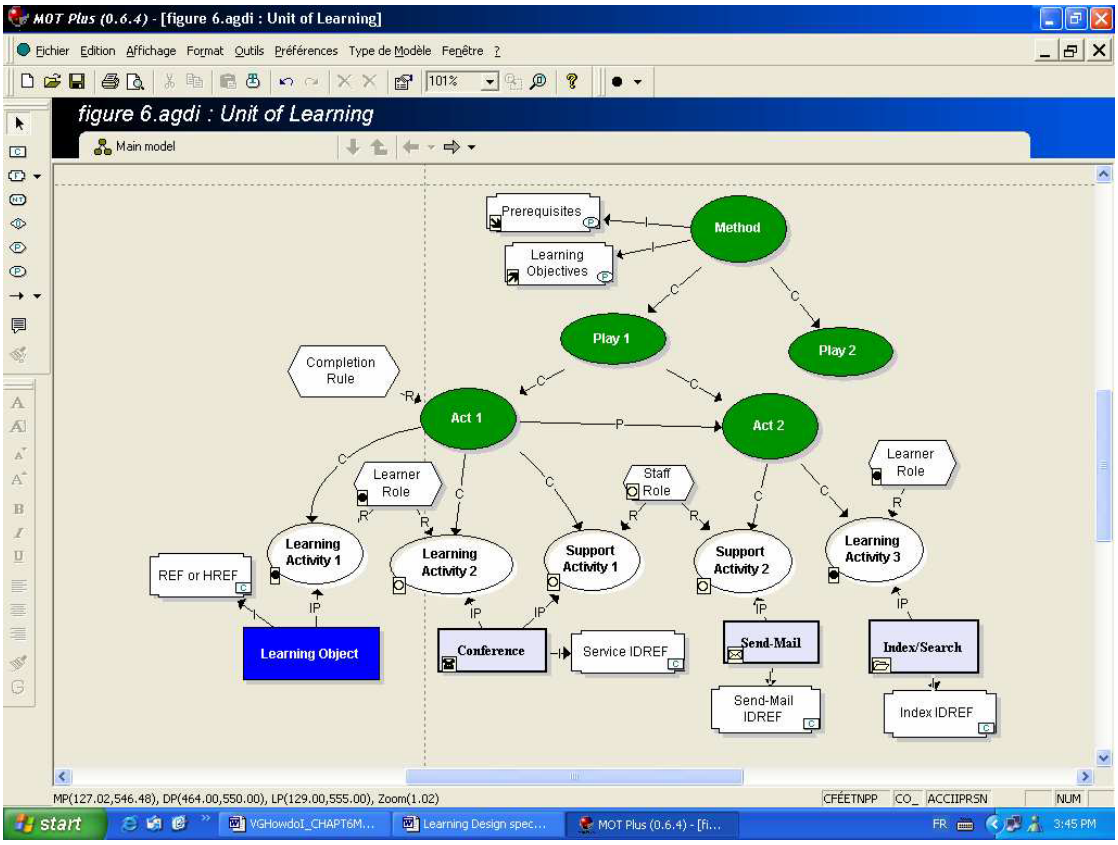


Figure 3.7: An equivalent MISA/MOT+ (MOT+LD) model to an IMS-LD example.

3.4 Blended Tools

There was also another very small group of LMSs that were especially good for PBL. They were designed and embedded with the content-oriented concept and the activity-oriented concept to meet the individual requirements of PBL. This section will discuss them further.

3.4.1 STELLAR

Derry et al. (2006) developed STELLAR, a course-development system that consisted of a collection of tools evolved through design research to support the creation, management, and study of online courses that intertwined problem-based learning with a reflective study of text and video. Derry et al. referred to STELLAR as "an experimental online PBL environment that enables teachers to engage with educational psychology concepts as they solved collaborative lesson design

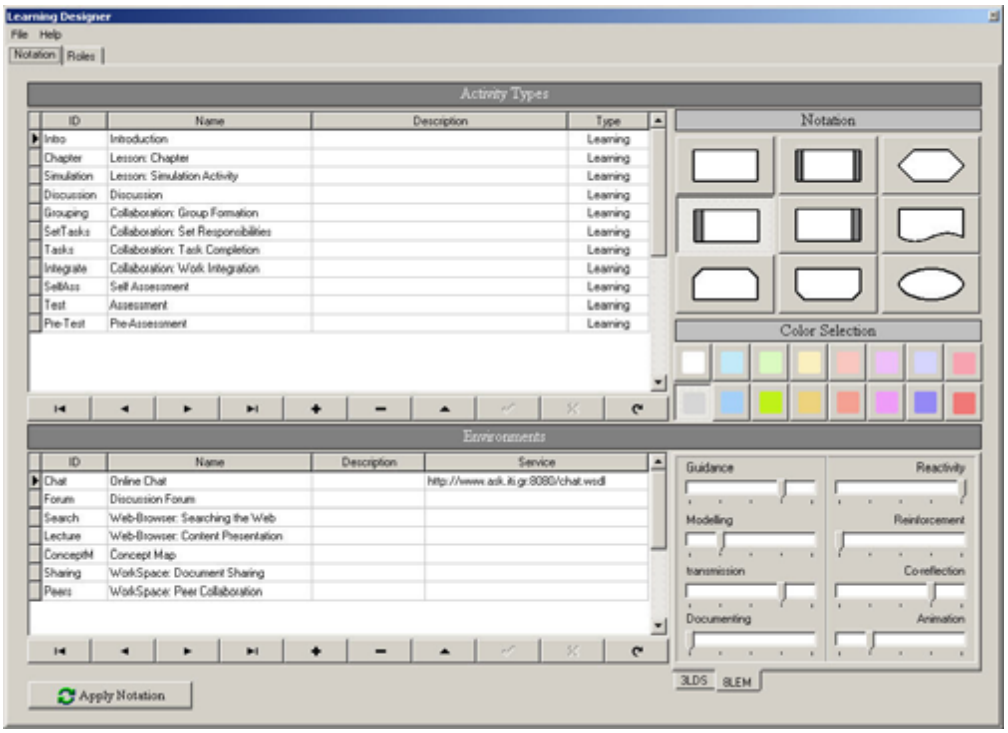


Figure 3.8: Supporting high-level notation in ASK-LDT.

problems that employed video cases as supporting instructional materials" (Derry et al., 2006; Hmelo-Silver et al., 2009).

As an important sub-system, STELLAR's PBL online module allowed PBL experts to create: (1) a library of video cases providing rich contexts for discussions and containing examples of classroom practice that could be used as materials for instructional problems, where the video cases took place when students engaged in the problems of design or redesign; (2) an online educational psychology hypertextbook comprising detailed explanations of course concepts linked to examples of those concepts found in video cases (the Knowledge Web), which helped students identify learning issues during problem-solving; and (3) a set of online instructional design problems, linked to selected video cases that followed a PBL format and provided the instructional activities for the course (Derry and Hmelo-Silver, 2005; Hmelo-Silver et al., 2009).

The design result later become a general script with steps for students to follow, as well as made available online tools, including personal notebooks, threaded discussions, and whiteboards, that students used in implementing the script. Figure 3.12 shows the video case linked to the knowledge web, and Figure 3.13 shows the complete learning steps of PBL. STELLAR was featured by providing a ca-

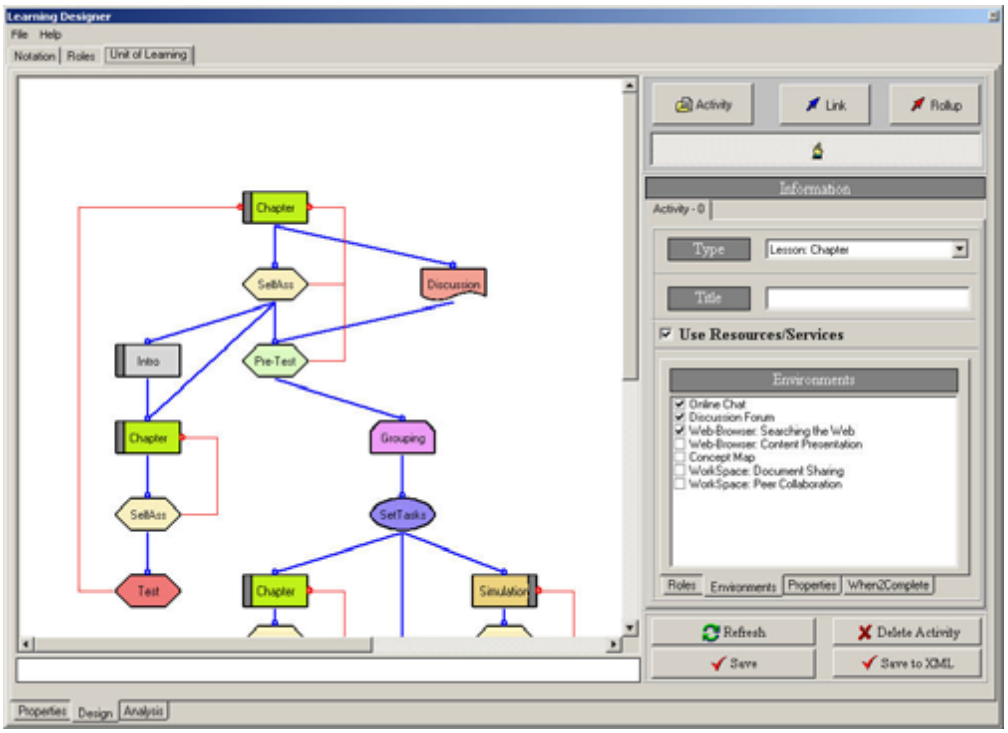


Figure 3.9: Supporting learning scenarios design and role definition through a graphical design interface in ASK-LDT.

pability that allowed a small number of facilitators to distribute their attention among multiple small groups (Hmelo-Silver et al., 2009), which was significant in implementing PBL in larger classes.

STELLAR provided a space to create an appropriate and relatively complete PBL design and implementation environment for a successful PBL.

3.4.2 Moodle Add-on: ePBL

Although there were many functional components in Moodle available for PBL to use, these components were not well organized regarding implementing PBL as a whole. Therefore an add-on was developed to support Moodle-based PBL.

ePBL was an add-on that extended Moodle to support PBL (Ali and Samaka, 2013). This extension, more specifically, a workflow inside the add-on, was designed based on Wood's PBL model (Woods, 2000). Therefore, the learning process controlled by the add-on was well defined. At any time, students were guided to work on the activities in a particular stage. Once all the deliverables, feedback, and subtasks were completed, students were allowed to move to the next

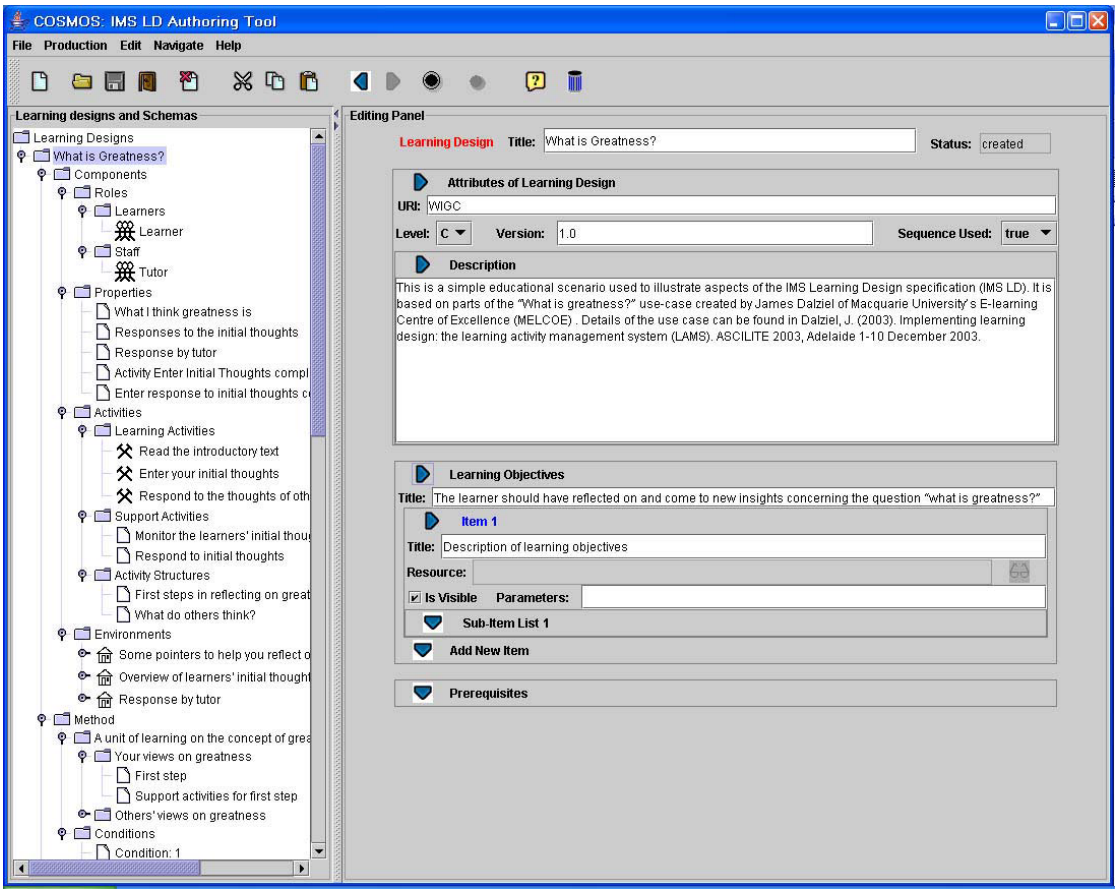


Figure 3.10: The User Interface of CoSMoS.

stage. Driven by a problem, students engaged in their tasks sequentially one by one. Each task had three stages: goals, teaching, and assessment. A facilitator administrated the task schedule. Figure 3.14 shows the multi-functional layout in ePBL. Figure 3.15a and 3.15b shows some other screenshots concerning the above working principle.

3.5 Summary

This chapter can be summarized into three aspects: (1) a comparison of the tools from the perspective of supporting and facilitating PBL, (2) a summary from the perspective of the tool's categories, and (3) a basic thinking of using emerging technologies.

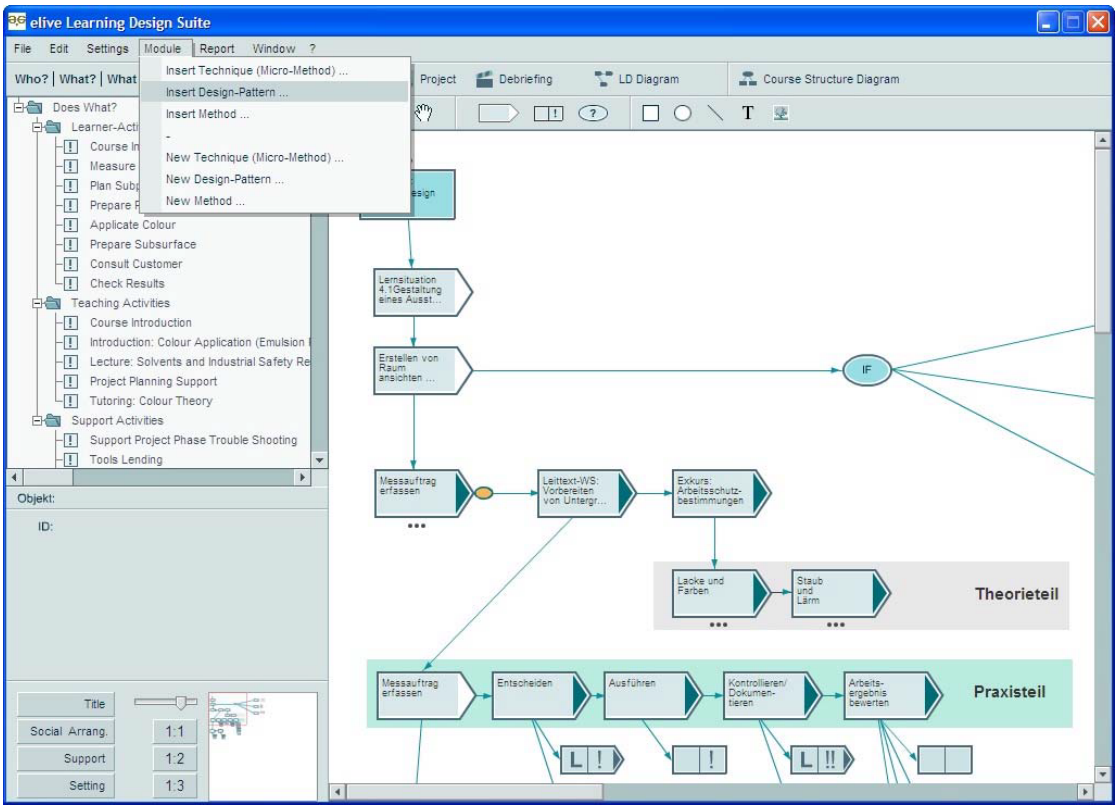


Figure 3.11: eLive LD-Suite: a screenshot showing the general function.

3.5.1 A Comparison of the Tools

In this section, a table (see Table 3.1) is provided for giving a comparison summary of the existing tools or systems. From the perspective of supporting and facilitating PBL, the comparison follows the following several criteria:

- **Provide the learning process authoring function for designing PBL plans**. This point was imperative because the design of the interactive teaching-learning process was a major difficulty in technology enhanced PBL practice. This has already been elaborated in Section 2.2. In Chapter 4, more details about this difficulty will be further presented and analyzed. In the comparison table, this criterion shortly as **Easy process authoring**.
- **Concepts and terminologies were close to the PBL daily practice**. As not all the tools were designed for teachers and students to use for PBL, so whether or not the concepts and terminologies provided by those tools were close to the PBL daily practice in order to reduce the difficulty to teachers to map their idea and express their PBL process design should also be considered. This criterion determines the **Designer type** of the

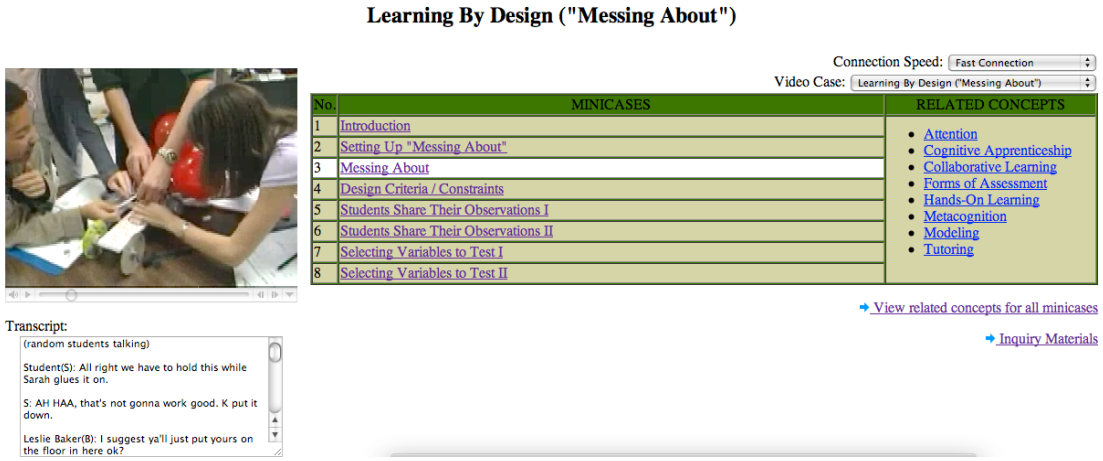


Figure 3.12: Video cases linked to the knowledge web in STELLAR.

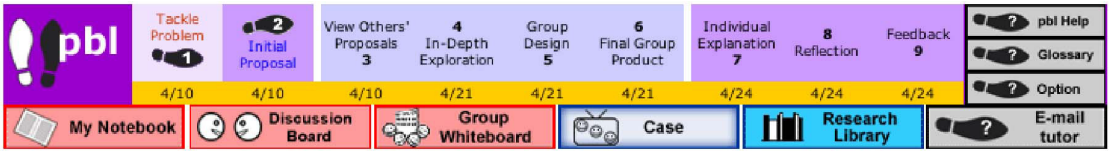


Figure 3.13: The concept of the STELLAR sidewalk.

PBL authoring. If concepts and terminologies were not close to the PBL daily practice, the designer might be experienced teachers, or even only PBL experts. If they were close to the practice, the non-experienced teachers could also use the software to design PBL processes. It is notice that, for the systems/tools who did not provide Learning Design authoring functionality (PBL process must be hard coded), the designer was the module developer.

- **Support the configuration of groups for the collaborative learning in PBL**. Group based collaborative learning is a very important feature in PBL. The configuration of groups should include the setting of the number

Figure 3.14: Problem workspace layout in ePBL.

(a) Issue phases in ePBL.

(b) Status of the required items in a phase.

Figure 3.15: Some snapshots about the working principle in ePBL.

of groups, the expected number of member for each group; the structure of certain group (e.g.: some groups may consist of several sub-groups); the specification of roles of the group members (e.g.: the role may be tutor, group leader, etc.) and the operation permission, or learning activity access policy, for these roles; etc. This criterion is shortly noted as **Easy group configuration**.

- **Enable to easily implement PBL plans**. Providing a learning process run-time function to facilitate teachers to implement PBL plans is also very important. Without an effective run-time function, a technology enhanced PBL plan can not be really conducted. The run-time function should include group/role population and a virtual learning environment to let all the learning participants to take part in. The group/role population indicates to assign teachers or learners to groups/roles for different PBL activities. The virtual learning environment should be able to group the teachers and learners and grant them the learning activity access permission appropriately (e.g.: in some scenarios, only teacher can create new learning text or other learning materials, but learners are read only to those text or materials; in some other scenarios, learners may be peer-to-peer, all of them have the right to create learning note, read and even modify other's notes). Besides, the the virtual learning environment should be integrate tools such assessment tools, communication tools, sharing tools, content management tools, cooperation tools, etc. This criterion is shortly noted as **Easy implementation**.
- **Provide the rich function of learning resources access and management**. The importance of this point is inherited from learning management systems (LMSs). Since educational organizations widely and successfully

adopted many LMSs for teaching-learning activities, this content-oriented based feature is also necessary for PBL. This criterion is shortly noted as **Rich resources management**.

- **Consider the contemporary user experience**. The contemporary user experience was a vague concept. It required a system to provide better performance (including low latency, high-speed transportation of data stream, and/or a large volume of storage) or achieving an overall capability for cross-platform (ranged from including different operating systems, Web browsers, to various types of mobile devices), or supporting complex and massive real-time collaboration, etc. In this comparison, I will use a general term, **Web-based**, to imply most of these features.

3.5.2 Types of Tools

Activity-oriented LMSs LAMS, OpenGLM and MOT+LD were the good choices for helping teachers organize and customize problem-based learning processes; however their common shortcomings were also apparent. That is, the notations provided by those systems were the general concept of learning activity rather than these specific terms frequently used in PBL practice. It was still difficult to develop PBL courses by using these tools. This problem also happened to the IMS-LD based tools. What was worse was that IMS-LD based tools commonly lacked good run-time tools, which could be another serious issue of the IMS-LD world. Although, compared to the content-oriented learning management systems, activity-oriented learning management systems were generally not that mature; it was considered that these systems represent the direction of development of the concept of Learning Design. Especially, the underneath theories and the corresponding implementations for IMS-LD could be considered to be an extremely important theory and application reference for the systematical support of the PBL.

Content-oriented LMSs Content-oriented LMSs were the most popular way to facilitate PBL. However they only provided a continuously improved "old fashion" way, meaning that was no advanced learning design concept embedded. In current PBL practice, it was common that teachers might neither had the appropriate expertise of learning theories and design methods nor had the thorough understanding of the potential capability of integrating different tools in a certain order correctly. Content-oriented tools or systems

only placed a rich set of tools in front the teachers and students, who both lacked guidance to design high-quality PBL plans and lack teaching-learning process control capability facilitating the implementation of PBL.

Blended LMSs STELLAR-PBL and ePBL relied on certain particular PBL model, which could be useful in certain circumstances but were limited to be applied to other situations. Teachers and students had to work by following the workflow implemented in the tool. Moreover, STELLAR-PBL and ePBL provided less or even no flexibility for teachers to customize PBL models to satisfy their individual PBL strategies.

LMS/Tool	Easy process authoring	Designer type	Easy group configuration	Easy implementation	Rich resources management	Web-based
Learning content-oriented LMSs, e.g. Moodle	Not applicable	Module developer	Yes	Yes (only when there was PBL module)	Yes	Yes
LAMS	Yes	Experienced teacher or PBL expert	Yes	Yes	Yes	Yes (Only latest version)
CopperAuthor (+ RELOAD LD Player)	No	PBL expert	No	No	No	Yes
RELOAD LD Editor + RELOAD LD Player	No	Experienced teacher or PBL expert	No	No	No	No
OpenGLM (+ RELOAD LD Player)	Yes	Experienced teacher	No	No	Yes	No
ILDE (OpenGLM + GLUE!PS System)	Yes	Experienced teacher	No	Yes	Yes	Yes
Mocolade (+ RELOAD LD Player)	Yes	Experienced teacher	No	No	No	No
ALK-LDT (+ RELOAD LD Player)	No	PBL expert	No	No	No	No
MOT+LD (+ RELOAD LD Player)	No	PBL expert	No	No	No	No
STELLAR	Not applicable (Hard coded)	Not applicable	Yes	Yes	Yes	Yes
ePBL	Not applicable (Hard coded)	Not applicable	Yes	Yes	Yes	Yes

Table 3.1: A comparison of the PBL applicable tools.

Chapter 4

Problems and an Conceptual Solution

Scholars such as Rob Koper and colleagues claimed that prevailing content-oriented learning within technology enhanced learning for instructional design pushed complex learning activities into a very narrow model that only provided digital content and virtual learning environments and ignored the richness of interactions between teachers, learners, resources and environment (Koper, 2001). This argument meant that the content-oriented approach was not efficient for those teaching-learning approaches embedding rich interactions and complex learning path. Consequently, according to the characteristics elaborated in Section 2.2, PBL was a relatively complex teaching-learning approach that emphasized giving an appropriate sequence for learning activities and encouraging the collaboration in learning activities, and that the content-orientation was inadequate for PBL. Therefore, the key consideration for us was that how to embed the concept of activity-oriented ICT supported learning design to provide a better way for the goal of facilitating PBL.

4.1 Problems with Current Technological Options for PBL

From the existing applicable tools, we knew that the content-oriented learning management systems might be not the right direction for the PBL from the starting point of consideration. Therefore, the problem analysis in this section will focus on

the other two modes: utilizing the learning activity-oriented LMSs and utilizing the blended LMSs.

4.1.1 Problems with the Existing Tools and Systems

From the previous chapters we knew that the insufficiency of current activity-oriented LMSs were as follows:

- The blended LMSs, such as STELLAR-PBL and ePBL, were designed relying on specific PBL models which were useful in certain circumstances but not applicable to other situations. Teachers and students had to follow the workflow implemented in the tools. Moreover, the blended LMSs provided less or even no flexibility for teachers to customize PBL models in order to satisfy their individual PBL strategies. This option was not the optimized way for our goal for PBL.
- The activity-oriented LMSs such as LAMS, Reload tool set, IMS-LD based tools, etc., were the right choices for helping teachers to organize and customize problem-based learning processes. Nevertheless, their common shortcomings were also obvious. The notations provided by those tools were the general concept of learning activity rather than the specific terms frequently used in PBL practice. For example, the vocabularies of IMS-LD were pedagogy-irrelevant and technology-oriented words such as *person*, *method*, *play*, *act*, *property*, *learning object*, etc., thus practitioners had difficulties of representing complex learning activities using the IMS-LD concept (Miao and Koper, 2007; Neumann and Oberhuemer, 2009; Griffiths et al., 2009). This meant there was difficulty in developing PBL courses with these tools.

4.1.2 Problems with the Modeling Standard

In addition to the two aspects above, there was another problem in the IMS-LD specification-implementation mechanism: a contradiction existed between the model that was defined by highly abstract concepts and the running specification that was needed for a computer system. Mellor et al. (2003) had also figured out a similar argument from programing language point of view. Figure 4.1 shows this kind of contradiction. The subject matter axis had an inverted scale with the abstract level of language. However, computers always needed enough detailed

information of subject matter which was not easy to thoroughly identify at the beginning in preparation for the final execution. It could be considered that eliminating this contradiction in a modeling-execution system required evolution and refinement, which was based on a phase cycle consisted of standard definition, digital implementation, evaluation, and standard improvement, to resolve.

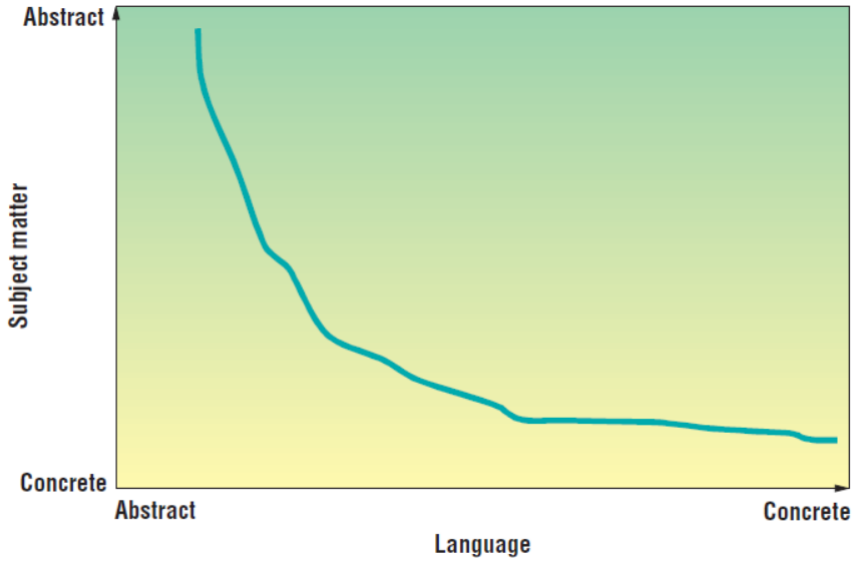


Figure 4.1: The relationship between a language's abstraction level and the abstraction degree of the subject matter (Mellor et al., 2003).

With the idea of specifying in advance a framework for capturing all possibilities in design teaching-learning activities, then the digital implementation works would follow, the IMS-LD specification defined a way to develop activity-oriented tools from top to bottom—meaning we first had the highly abstract modeling standard, then developed systems providing Learning Design functions based on the standard. However, the contradiction stated in previous paragraph exposed problems and showed considerable risk to this way from two ends:

- At the top end, the modeling standard (or modeling language), especially with such an ambitious goal of providing a general learning expression for all the main educational approaches, was not easy to be mature.
- At the bottom end, the content of a learning model, especially with such a comprehensive representation of reflecting the complex controls and logics in collaborative learning activities, was not always technically implementable.

The existing researches and practices implied those two problems. Following this top to bottom way, there was a number of IMS-LD based or compliant tools

developed. Those tools included Reload tools, ASK-LDT, SLED, CosMos, eLive LD-Suite, etc. However, most of those tools were undergoing maintenance or had even disappeared. While some of these tools or systems were still available, they were rarely really used in educational practices. Additionally, while most of the tools were the authoring tools, only a few provided run-time function or environment.

More specifically, from the viewpoint of teaching-learning activities:

- Teachers didn't feel that the authoring tools based on IMS-LD provided enough function or concept components to describe their teaching-learning process.
- The concepts from IMS-LD (entity notations, entity relationship settings, and attributes) were not close to their specific teaching-learning needs, thus teachers didn't feel the Learning Design was easy to perform (this problem similarly happened to LAMS).
- There was no good run-time tool that to finally facilitated teachers and learners for PBL implementation.

From the viewpoint of technical implementation:

- It was indeed difficult to develop the run-time tool to interpret the teaching-learning model defined in IMS-LD with a focus on general educational purposes.
- There was considerable risk in that if there was any imperfection of the standard, for example, some concepts needed to be deprecated or added, or even if the concept structure needed to be redesigned, modified or any other revisions required, then all the implementations which were based on the standard would encounter fundamental problems or even failure. In summary, boldly speaking, the unpopularity of IMS-LD authoring tools implied that the modeling standard was far from mature, and it was rare to see an IMS-LD run-time tool that proved that the modeling output was not generally technical implementable.

To summarize, the contradiction was that, if a model was designed to be able to easily represent different learning processes for none technical teachers, then the ICT system which was able to interpret the corresponding model instances tended to be difficult to implement. Or if a model was designed to implement, then the

model would be different to let none technical teachers to use.

4.2 Between Domain Semantics and Generality

Consequently, two questions were raised: (1) Regarding the theory, was it really possible to define a generic modeling standard to cover all, or at least most of all, the different learning activities, especially when they usually consisted of extremely complex collaborative activities? (2) Concerning the technology, was it really possible to define so many different logics, exceptions and collaborations using such a high abstract modeling language, especially when there were still difficulties to describe and control the logics and collaborations using such low abstract languages such as Java, C++, etc.?

If we wanted to model a teaching-learning activity with both broad and fine enough expressiveness, the modeling language needed to have an abstract level low enough. Otherwise, a modeling language defined by a high level of abstraction has either the expressiveness reduced or the outcome model was difficult to be computer processed. From this angle, we could require a lower abstract modeling language for problem-based learning design. However, this created a new problem, which was that a low abstract level of modeling language needed more knowledge and skills of modeling. It was clear that the targeting users were ordinary teachers, making the additional modeling knowledge and skills demand unreasonable. Therefore, instead of using an education-generic modeling approach for PBL, it might be better to find a trade-off learning design approach which aimed at to be optimized for PBL, where the trade-off should be at the balance point of the following criteria:

- The abstract level of the learning design modeling language needed to be as high as possible, but only for the purpose of representing entity notations, relationship settings, and attributes in PBL domain, in order to enable teachers easy to use.
- The learning design modeling language needed to be as general as possible to represent the most different mainstream PBL models or patterns, in order to provide a flexible enough modeling capacity (or high enough expressiveness).
- The representing entity notations, relationship settings, attributes, terms, etc. of the problem-based learning design modeling language needed to be as close as possible to the practice of PBL.
- In the modeling language, there needed to be some collaboration patterns

that could also generally represent all the collaborative activities in PBL. In this way, teachers would be able to design the collaboration for PBL more easily; however, which might be more significant, the difficulty of developing run-time tools could be reduced, since collaboration activities were not that unpredictable.

Facing the criteria above, an important fundamental work needed to be introduced. That is, Miao et al. (2015b) had proposed a concept of a PBL scripting language.

4.3 PBL Scripting: A PBL Ontology Modeling Language

This section is particularly based on the publication "Using a PBL Authoring Tool to Train Teachers in Designing an Online PBL Unit" by Miao, Y., Samaka, M., Wang, D., Ali, Z., and Romanowski, M. (Miao et al., 2015b).

4.3.1 Basic idea

Analogous to the music notation that contained enough information to convey musical ideas from one person to another over time and space, researchers in the field of learning design tried to develop a notational system for describing and sharing learning design ideas. Many learning design languages and associated tools had been developed in the past decade such as IMS-LD (IMSA, 2016), LAMS (Dalziel et al., 2003), LDL (Martel et al., 2006), and CompendiumLD (Conole et al., 2007). These learning design languages were developed for describing a wide range of pedagogical strategies.

However, researchers had argued that it was almost impossible to develop a new generation of e-learning environments that were completely pedagogically neutral (Laanpere et al., 2012). In order to support the design and sharing of PBL practices, Miao et al. (2015b) proposed that it was needed to develop a PBL specific leaning design language, named PBL scripting language.

According to the schema theory (Schank and Abelson, 2013), generalized knowledge about a list of the characteristic events involved in a common routine was called a script. Scripts could be used to organize procedural knowledge, to assist

recall, to guide behavior, to predict likely happenings, and to help individuals make sense of our current experiences. People knew how to behave and what to expect in particular situations by using scripts. Scripts were mental structures representing the person's knowledge about objects, people, or situations. They were derived from prior knowledge and experience, and they set up expectations about what was probable and appropriate in relation to particular social contexts.

Figure 4.2: A typical restraint script.

For instance, figure 4.2 shows a diagram that illustrates a script of a typical restaurant, in which the process of eating at a restaurant was divided into five scenes: sitting, ordering, serving, eating, and paying. When a scene finishes, another scene started. In this restaurant script, there were three roles: consumer, waiter, and chef. The script embodied knowledge about how people in particular roles (e.g. waiter, or customer) are expected to behave in each scene. For example, it was expected that a chef prepared the food that the customer ordered and a waiter passed the food to the customer in the serving scene. After being served, the customer ate the food in the eating scene. Such expected behaviors were called behavior rules. A behavior rule combined a role, an action, an object, and others such as tool and resource. For example, a chef cooks the food ordered by the customer by using a frying pan and meat. The people with a certain role followed the behavior rules in the social interaction. Violation of behavior rules created trouble. For example, the waiter ate the cooked food or the chef prepared food that was not ordered by the customer. In order to represent procedural knowledge in restraint, we could define vocabularies and relevant rules that can be used to specify various restaurant scripts such as those used in McDonalds or a buffet style restaurant.

To define the PBL scripting language, Miao and colleagues had referenced the framework of the aforementioned schema theory. Like the restaurant knowledge, teachers' knowledge was situated, event-structured and episodic (Putnam and Borko, 2000). Similarly, teachers' planning was situated and contextually sensitive (Sardo-Brown, 1990). Yinger (1979) asserted that teachers' planning could be characterized as decision making about the selection, organization, and sequencing of routinized activities. Therefore, it was feasible to use a script to represent

teachers' knowledge and planning. In the CSCL research community, the concept of the script also had been used to designate scaffolds structuring face-to-face collaboration and computer-supported collaboration (Dillenbourg, 2002).

As a basic idea, PBL scripting language was from the light of schema theory. In order to support technology enhanced PBL, the PBL scripting language was used to model computational descriptions to the problem-based learning processes. From this point of view, the PBL scripting language was also a domain specific language (DSL), where the domain is the PBL pedagogy and the language was developed through analyzing and summarizing common characteristics of well-known PBL models and the best PBL practices on the Internet. Furthermore, since this language provided an explicit specification of the conceptualization of problem-based learning (Gruber, 1993, 1995), we could also call it a PBL ontology modeling language.

4.3.2 Significance

By adopting the PBL ontology modeling approach, this scripting language seemed to be a good solution to answer the thinking of the trade-off in Section 4.2. The reason was: rather than providing general-purpose modeling conceptualization terms like the unified modeling language (UML) used for a wide variety of purposes across a broad range of domains, the PBL scripting language, provided only the vocabularies and the rules that selected directly from the PBL domain and optimized for the representation of the problem-based learning design problem. An the one hand, since the work of abstracting domain specific concepts, terms and rules was usually completed only by domain experts (Van Deursen et al., 2000), and experienced technical experts developed domain-specific description generators and transformation definitions, the quality of problem-based learning design could be ensured; on the other hand, since the concepts, terms and rules were those things that educational practitioners, even for less experienced, were familiar with, the ease of user of the problem-based learning design could also be foreseen.

Besides, according to Noy et al. (2001), in general, developing an ontology of a domain could also proved the following benefits: to share common understanding of the structure of information among people or software agents; to enable reuse of domain knowledge; to make domain assumptions explicit; to separate domain knowledge from the operational knowledge; to analyze domain knowledge.

It is noticed that, educational modeling languages (EMLs) and IMS-LD were also domain-specific modeling languages in comparison to the unified modeling language (UML). However, the domain of EML and IMS-LD was the entire educational profession aiming at supporting a very wide range of pedagogic strategies. On the contrary, in PBL scripting language, the term domain indicated the PBL strategy only.

It is also worthwhile to mention that reusing and customizing Collaborative Learning Flow Patterns (CLFPs) (Hernández-Leo et al., 2006; Villasclaras-Fernández et al., 2009) was also a potential effective approach to help non-experienced teachers to create their own particular yet professional problem-based Learning Design. However, the main limitation was that the pattern-based approach was still not flexible enough for some PBL practices. For example, this approach lacked to support the requirement that if a teacher wants to combine two different well-known PBL models/patterns to form a special learning process to suit her special teaching-learning situation. While using the PBL scripting language, the dynamic combination was achievable, since the ontology based approach has a higher abstraction which provides a higher flexibility in process modeling than using the predefined processes in the pattern-based approach.

The following sections will explain the ontology conceptualization of PBL as a systematic account of structured interpretation that educational practitioners use to think and communicate in the environment problem-based learning.

4.4 The Syntax Structure of the PBL Scripting Language

This section is based on the publication "A Domain-specific Modeling Language Approach to Support Various Forms of Online PBL" by Wang, D., Miao, Y., Hoppe, U., and Samaka, M. (Wang et al., 2014b)

The last section introduced a concept of the PBL scripting language from a general way. Barrows (Barrows, 1986) proposed a taxonomy to classify different methods of PBL. He differentiated several forms of PBL according to (1) the complexity of PBL problems, (2) the focus on teacher or student-centered learning, and (3) the order of problems, cases, and information presentation. Ryberg et al. (2010) suggested a way differ PBL among different constructions by distinguishing how

control or power was distributed between teachers and learners. This way was analyzed through three dimensions including the problem, the work process, and the solution. Based on their ideas, we have further researched the variety of PBL from a perspective of process modeling.

In this section, we will look into a concrete PBL model, introduce how a PBL process model was usually described informally in natural language and explain how a model is implemented in PBL practice in the case of trying to find the common syntax structure of the PBL scripting language from different PBL models.

4.4.1 PBL Use Case

As it was introduced in Section 2.2.3, there was no universally agreed upon set of practices to guide the implementation of PBL, and as it was mentioned in the last section, depending on different learning contexts, PBL could be conducted in a number of ways based on different models and practices. Those models and practices included the McMaster PBL model (Woods, 1996), the Maastricht "Seven jumps" framework (Gijselaers, 1995), Woods' model (Woods, 2000), Mills' five steps model (Mills, 2006), Seymour's "five-stage" model (Seymour, 2010), SALFORD model (McLoughlin and Darvill, 2007), etc. as well as some well known practices published on the Internet, such as IMS (2003); IMSA (2016); Gulfcoast (2016), etc. For instance, among these models or frameworks, a relatively popular one was the Maastricht "Seven jumps" model. It was designed as a framework to facilitate and structure students' problem-based learning processes originally at the Maastricht University (Gijselaers, 1995; Barrows, 1996). The following is a description of the "Seven jumps" model taken from Maurer and Neuhold (Maurer and Neuhold, 2012):

> 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. 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 en-

sured 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—**brainstorm**. The rationale behind this step is that students collect potential interests that they might have, activate prior knowledge, and share certain expectations. 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). Just in case a group member does not understand how a 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. 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. This is the most challenging step for inexperienced 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 way of formulating learning objectives in the ideal case reflects the different approaches to the wider topic that students have agreed to research upon, because they consider them to be the most relevant to the specific topic and because they are interested in exploring exactly these questions. Additionally, by agreeing on common learning objectives in a group, experience showed that students also get acquainted to formulate learning objectives clearly and to the point, as otherwise the post-discussion in the tutorial group

> goes into too many different directions.
>
> After these five steps of the pre-discussion, students leave the group again to engage in the **self-study**, which takes a central position in the PBL framework and emphasizes the self-responsibility of the learner for knowledge acquisition. During this self-study students should work on their individual answers to the formulated learning objectives. Especially for students in their first year of study the key literature is provided after each assignment, while this should not discourage students to look for additional sources and other literature that they might find interesting. For more advanced students, sometimes just a general reading list for the whole course is provided, and it is up to the students themselves to decide in their self-study, which of the literature provided is relevant for their respective learning objective. Students thereby also learn how to select relevant material and literature in a relatively short period of time. The following tutorial, normally taking place two or three working days later, starts with the **post-discussion** where students report back, exchange their answers, discuss problems and try to come to common conclusions of how to answer the learning objectives. While students should be able to come to a common understanding of some relevant factual knowledge during this post-discussion, it is especially the more normative and not-straightforward answers that then allow for a more profound discussion and exchange of arguments. By experiencing different perceptions of a question by their peers, by listening to different lines of argumentation, and by being confronted by different perceptions of perhaps the same reading, students are acquainted to report, listen, discuss and debate.

As the name implied, and as it was found from the previous quotation, this PBL model consists of seven steps, which included *Clarification of Terms and Concepts*, *Formulation of Problem Statement*, *Idea Brainstorming*, *Categorizing and Structuring Ideas*, *Formulation of Learning Objectives*, *Resolution through Self-study*, and *Conclusion by Peer Evaluation*. By examining the first step, *Clarification of Terms and Concepts*, it further consisted of three activities: *Reading Assignment*, *Discussing Unknown Words or Concepts*, and *Understanding the Text as It Stands*. This layout similarly appeared to other steps. Generally speaking, each step consisted of several activities; each activity would be performed either by the facilitator, individual students, student groups, or by other stakeholders, e.g.,

scientific staff.

However, this was just a model or a framework rather than an implementable instance, meaning that teachers still needed to full it with specific information and settings before it could be implemented. We call this task **instantiation** and call the result of the instantiation **instance** in the rest of this thesis. Sometimes, this model even needed to be extended and modified to fit certain specific learning contexts. For instance, in practice, teachers needed to decide how these steps and activities would be actually arranged, because it depended on concrete situations, such as the number of students, the learners' prior knowledge, the group structures, the problem used to drive learning, the topics to be learned, the availability of learning technologies, etc. Therefore, even with the help of the model framework, teachers probably still felt different to design a PBL plan. Additionally, there were so many different models to choose from, teachers could had been confused when deciding which one was better.

As an instance of the "Seven jumps" model, we worked out a blended PBL scenario called "Deformed Frogs (Linn et al., 1999)," which was originally from WISE (Web-based Inquiry Science Environment) (Linn et al., 2003) library projects for a long time as a chemical hypothesis arguing frog deformities were being caused by an environmental chemical that stimulated growth. This instance will be the example referenced throughout this thesis. In this PBL instance, we assumed that the PBL course was conducted in a classroom with a digital whiteboard at the front and every student had a separate PC. The students involved in this scenario were divided into several small groups. The learning process started with a facilitator giving reading assignments and materials about the discovery of deformed frogs in a local area to students. This challenged the students to investigate the status of the frog population and encouraged them to take a proactive stand on this environmental concern. Then, the facilitator coached the students to identify and understand the problem. After a discussion, the students identified the problem: "what is the cause of deformity of the frog and how to prevent it from spreading?" The students identified major issues connected to the problem. The identified issues were frog habitat, the various types of deformities in frogs, wetlands, watersheds, the effects of pollution on a natural habitat, and so on. In this scenario, the students acquired knowledge through presenting, arguing, and evaluating the hypotheses and solutions.

Figure 4.3 depicts the structure of the PBL scenario described above. The left part of the diagram presents a sequence of steps as the overall process of this scenario. In our work, we called each step a PBL phase. The right part of the diagram illustrates the internal structure of the first phase. In this phase, students in the class found the problem by reading assigned learning materials at first. Then, they discussed the unknown words or concepts in groups. Each group created a list of unknown words or concepts. Finally, the facilitator helped students to clarify the unknown words or concepts and to produce notes as the learning outcome of the phase. All these activities were carried out by using the digital whiteboard. Some activities produced artifacts or needed learning resources.

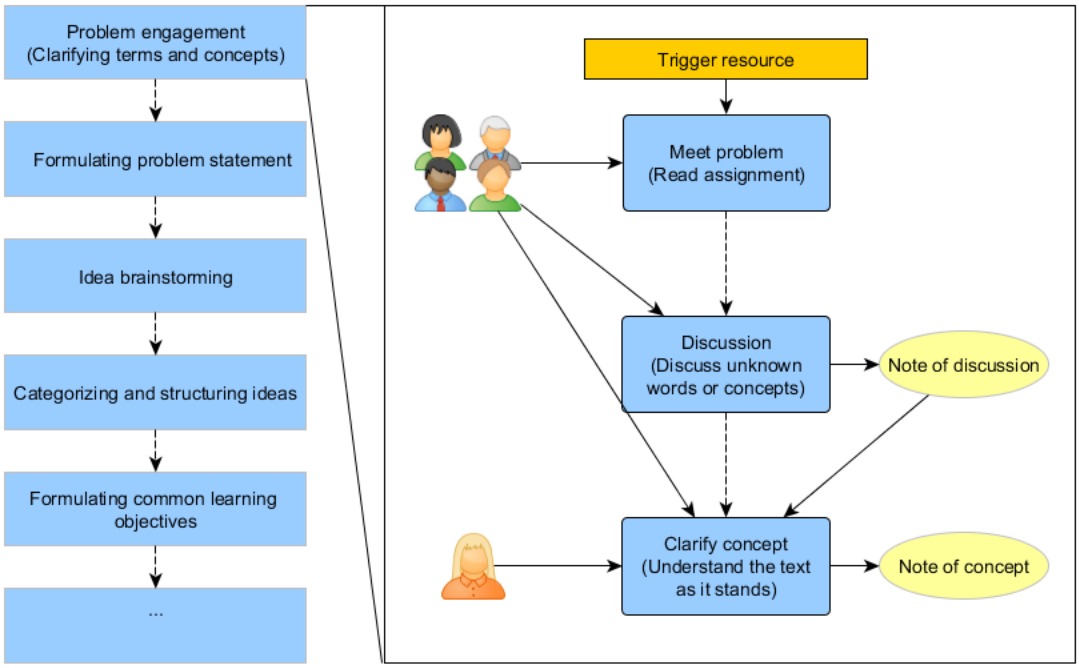


Figure 4.3: The process structure of the PBL scenario "Deformed Frogs."

In fact, after a broad review of those models, we found that there was actually a common pattern that appeared in all of those models. This will be discussed in the next several sections.

4.4.2 Three-levels Structure of PBL

After our research, the various forms of PBL could be characterized by a common three-levels structure: they are a general description level: *Script Level*; a middle process level: *Phase Level*; and a detail description level: *Activity Level*.

- **Script Level**: The PBL environment should facilitate a whole PBL process that could consist of different phases with different sequences to provide a general overview.
- **Phase Level**: The PBL environment should facilitate a phase that could consist of different activities with different sequences to provide a detail logic.
- **Activity Level**: The PBL environment should facilitate an activity that could be performed by different actors individually or collaboratively, produce different artifacts, and use different learning resources and application tools, etc. to provide detail settings.

Script Level: A PBL method could be characterized by its constructions and their corresponding sequence. The constructions were also called steps or stages. In our work, we unified the terms and called them phases. For example, problem engagement, problem definition, identification of learning issues, research aim, investigation, generation of solution, evaluation, etc. were all the phases. We observed that McMaster's model consisted of six steps; Maastricht' model had, as the name indicated, seven steps; the Mills' model was made up of five stages. Different PBL models not only differed from the number of phases, but also from the aims of different phases and the order of phase sequences. For example, while the "Seven jumps" model encouraged generating tentative solutions before the study, most PBL models suggested generating solutions after acquiring knowledge in relation to research questions. In summary, PBL tutorial processes could vary from script level with different phases and their orders.

Phase Level: The goal of a phase could be achieved through performing different learning activities under different sequences. For example, in a phase of problem engagement, a problem could be introduced by presenting a case, describing a scenario, observing a phenomenon, or discussing a situation. For setting a learning goal, the third phase of the Mills' model consisted of the following activities: (1) formulate the key research problem, (2) identify the knowledge, (3) define three specific research tasks, and (4) agree on how the group will work together. In the "Seven jumps" model, the fifth phase was organized in the following activity sequence: (1) formulate learning objectives; (2) group reaches consensus on the learning objectives; and (3) tutor assesses learning objectives. Obviously, PBL course processes varied at phase levels with different activities and different sequences. Additionally, the sequence structure and the activities combined described the process

logics for each phase and was the most important implementation logic role for a PBL.

Activity Level: An activity could be arranged and performed in different ways. For example, for designing an activity "making a learning plan," there were many questions for making each following decision. Who (e.g., the facilitator or/and the learner) were the actors of which activity? Was an activity performed individually or collaboratively? What artifact (e.g., a learning action or a learning plan) was produced? What learning resource (e.g., a Web-page or a table) and a learning tool (e.g., a brainstorming tool or a Wiki) will be used? How long did an activity take (time-based or condition based), and how should it start (e.g., manually start or when certain learning resource was available) and finish (e.g., time was over or the tutor decided)?. Answering these questions differently in the activity level meant different designs and arrangements were configured.

4.4.3 Syntax of the Language

According to the three-level structure of PBL, the abstract concepts could be summarized as the following. The *Script Level*, as the highest-level of abstraction of the PBL scripting language; the primary PBL abstract phase was those such as *Problem Engagement*, *Problem definition*, *Knowledge identification*, *Aim*, *Plan*, *Investigation*, *Information sharing*, *Reasoning*, *Problem resolution*, *Evaluation*, *Reflection*, *Report*, *Application*, and etc. Each of the primary abstract phases was defined in terms of attributes such as *Title*, *Goal*, *Outcomes*, *Activities*, etc. that could be performed to achieve the outcomes. Additionally, we defined common abstract phases, such as *Management*, *Facilitation*, *Assessment*, *Communication*, *Collaboration*, *Co-decision*, etc., which could be associated with a primary phase. A PBL script consisted of a sequence of phases. Each phase could be defined by combining two or more abstract phases. The number of phases and their sequence could be decided by the designer of the PBL strategy. As a consequence, various forms of PBL could be designed by making different choices, combinations and sequences.

In *Phase Level*, each abstract phase, including the common abstract phase, had a list of possible activities that could be performed to achieve the outcomes. For example, the activities in phase *Investigation* included *Identify learning source*,

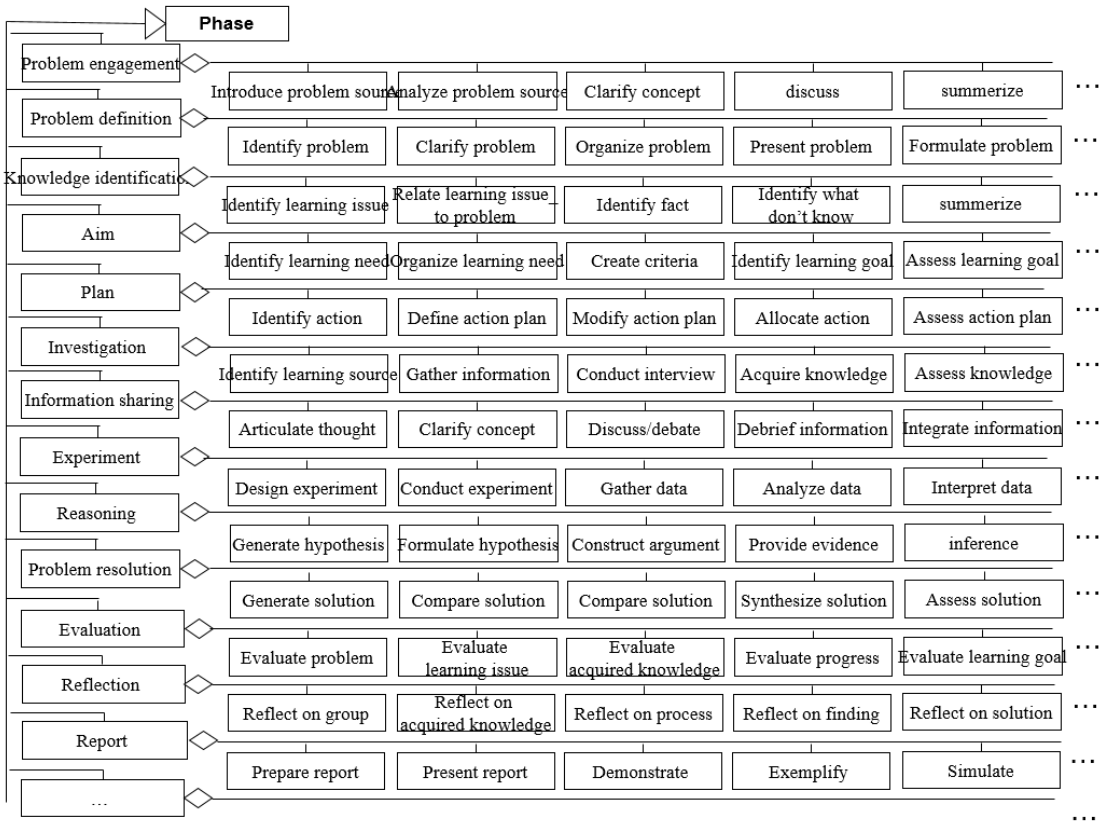


Figure 4.4: The abstract concepts of the script Level and the phase Level of the PBL scripting language.

Gather information, *Conduct interview*, *Acquire knowledge*, *Assess knowledge*, etc. Those activities were selected as work steps, being linked in to different sequences, to form a workflow of a learning phase. Under this context of an abstract phase, this level of script allowed PBL teachers to develop executable learning processes with no familiarity with formal programming concepts and help them more easily identify or ignore necessary or unnecessary activities. In this level, a single student, a group of students, or several groups could be clearly associated to one or multiple learning activities. As a result, this made teachers be able to flexibly define different learning activity paths for different PBL strategies. Figure 4.4 shows these abstract concepts of the script level and the phase level of the PBL scripting language.

In *Activity Level*, an activity could be performed in different ways. It could be assigned to all students in a class, to all students in each group or in a specific group, and even to a specific student. Certain activity could be assigned to the facilitator. An activity could produce artifacts such as learning objectives and

learning actions. Performing an activity could need some learning resources such as the information on a given Web page or a file, and application tools such as a brainstorming tool, chat tool, voting tool, wiki, and whiteboard. Some activities could be performed in an individualized organization where the selected actors worked on their own, whereas others promoted a certain level of social organization among actors or more strongly tied collaborative dependencies such as group work. In collaborative work, they could have different distributions among tasks and learning resources. They had different choices to use communication tools and collaboration tools that fit their teamwork. Therefore, various organizations and technologies could be used to conduct various forms of PBL.

Based on the modeling structure and the relationship among the elements inside the structure above, an abstract syntax of this language was initially proposed. Figure 4.5 illustrates the syntax structure. From this syntax, we could see that the center concept of the PBL scripting language was the activity. Activities constituted a phase. A phase followed another phase. People in different role performed an activity. An activity could produce artifacts, and vice versa, an artifact could be used in different activities. Service facilities, such as online brainstorming tools, and information resources, such as learning materials, could be also used in different activities.

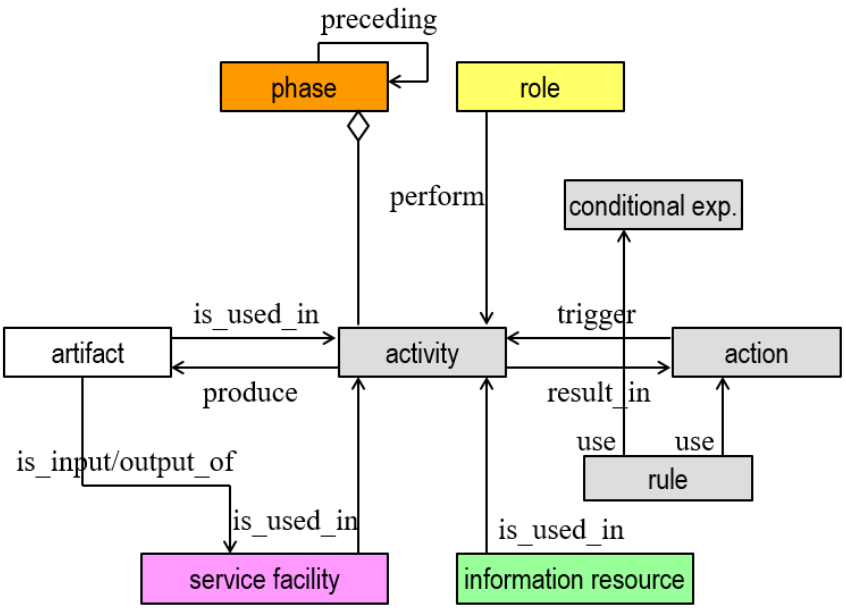


Figure 4.5: The basic entities of the PBL scripting language.

4.5 Summary

This chapter firstly discussed the problems of current technology-enhanced options for PBL. We found that the existing systems or tools either only provided less or even no flexibility for teachers (learning designers) to customize PBL models in order to satisfy their own PBL strategies or only provided general concepts of learning activity rather than the specific terms frequently used in PBL practice, which made it difficult for teachers to use.

Secondly, since IMS Learning Design specification and the upon systems or tools, which aimed at the facilitation of activity-oriented teaching-learning, provided a relatively complete experience ranging from theoretical research to technical implementation, and since facilitation of activity-oriented teaching-learning was also the vision of project PLATE, we additionally analyzed the problems of the IMS Learning Design "ecosystem." After this analysis, we recognized that, in the case of facilitating PBL, relying on a pedagogy genetic modeling language approach might not be an appropriate solution. On the contrary, we claimed that a PBL domain-specific modeling language approach should be considered. As a fundamental work for this approach, we introduced the PBL scripting language. Then through a PBL model and instance, we introduced a three-level structure of PBL, further showing the syntax of our PBL domain-specific modeling language.

This chapter presented an ontological solution to the problem-based Learning Design, which provided an essential research result for designing a new generation of ICT enhanced problem-based learning environment. However, this ontological solution is just a fundamental idea. The technical design to implement the ontological solution should be seen as the core challenge of the author's work for realizing the PLATE system. The next chapter will present how this technical solution has been sought.

Chapter 5

A New Approach of Model-Driven Learning Design

Based on the recognition of the problems of current technology-enhanced options, the problems of the IMS Learning Design "ecosystem", and the adoption of PBL domain-specific modeling language approach, a list of foreseeable technical requirements should be listed in the chapter. Those technical requirements were only the low-level functional problems. To meet those requirements, a high-level innovative system architecture design will to be proposed later in this chapter.

5.1 Functional Requirements and Architectural Challenge

In the previous chapter, the theoretical background of applying Learning Design to PBL has been analyzed, existing problems have been found, and as an initial approach to resolving those problems, a PBL scripting language had been initially formalized. However, the PBL scripting language approach was only an idea at the concept layer, aiming at the goal of realizing a software system to facilitate the implementation of PBL; there were still many concrete functional or non-functional technical requirements in the system design and the implementation layer needed to be overcome.

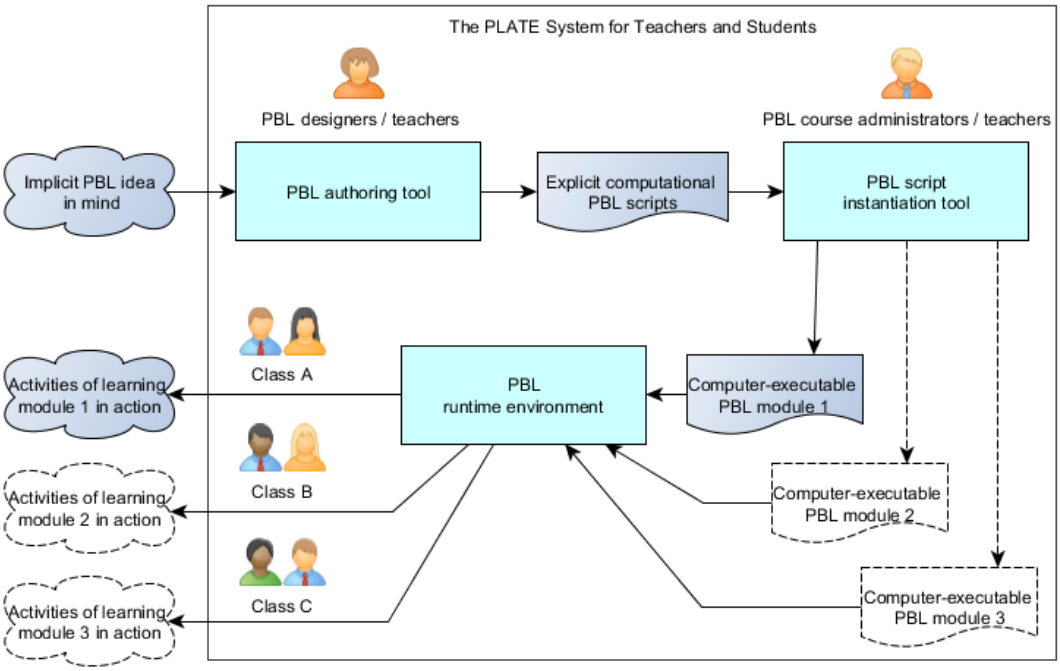


Figure 5.1: A use case of the PLATE system for teachers and students.

5.1.1 A PBL Visual Authoring Tool

The basic idea behind the technology enhanced PBL was Learning Design. Thus it also needed a virtual design tool for PBL process description. As a new tool in the 21st century learning context and making use of the power of the PBL scripting language, the tool needed to provide the authoring of three types of functional features: **flexible**, **responsible**, and **Web-based**. More specifically, under these three aspects of features, a number of major concrete functional requirements were listed as the following:

Having an Intuitive Authoring User Interface : As the review of the existing learning design tools shows, it seemed that a flowchart diagram user interface was the best choice for supporting learning design. In fact, there were many benefits of the flowchart diagram. According to edrawsoft (2016), the benefits of this kind of user interface included: (1) simply clarifying progress offered an easy, visual method to help involved members instantly understand what they should do step by step; (2) the designer could easily understand the meaning based on the rapid cognition of shapes or edges while finding out which step was unnecessary and which progress should be improved; (3) the revision operation such as removing unnecessary steps or

adding new steps in a script was very easy; and 4) the problem was broken into easily definable parts.

Supporting the Design of Role and Organization Structure : As it was described in Section 4.4.1, there should be a stage for the definition involved roles and organization structure in order to map the learning group or class. The function for this definition should be flexible enough to cover most different PBL participant settings. For example, the functions needed to have have grouping, membership authoring, resource access control, operation permission setting, etc.

Associating Role to Different Senarios : The authoring function in the learning script design stage should provide a rich range of capabilities that enabled the learning designer to specify the roles for multiple participants, where each role could engage in different activities simultaneously in a learning design.

Reusability of the roles and the organization structures : In PBL practice, usually the learning organization structure for a facilitator was not always changed. A facilitator might only have several fixed organization patterns and fixed roles defined inside. Therefore, for the reason of providing more convenience for the PBL design, the system needed to provide the reusability of the roles and the organization structures. The challenge was also to maintain the referential integrity.

Providing Guidance for PBL Course Design : Guidance was another important meaning of facilitating problem-based learning design. To achieve this function, the system had to provide a way of directing designers to model a sound PBL teaching-learning plan through a right path. The directing needed to consisted of two aspects. The first one was restriction. For example, only certain activities could be located in a certain learning phase, or although there was a flexible diagram user interface, some elements were not allowed to have an association with some other notations (or elements). The other aspect was that the system should intelligently give recommendations at the right time. For instance, when a designer created a notation to represent a learning activity, the system needed to prompt the design which tools could be used to this activity or which roles could be associated with this activity, etc.

Handling Branch in Activity Sequence : Britain (2004) mentioned that a

successful facilitation of PBL was not just to enable the creation of thoughtful activities and engaging students to undertake activities, but also to involve making setting to the sequence and timing of the various activities easy. This orchestration could form a simple sequential flow in most cases. However, sometimes a Learning Design also involved a branching of workflow into parallel activities undertaken by sub-groups before coming back together. Or a learning process could be constructed that allowed different routes to be taken based on the achievement at a testing stage within a sequence. Thus a key aspect of the tools to support the concept of Learning Design or the problem-based learning design would be the notion of the workflow.

Supporting the Online Feature : Since the target users of our system were those people who had been familiar with the 21st century Web-based application, and since there were a lot of benefits when a system was Web-oriented, and also since Web-based systems were the mainstream of current Internet applications, we needed to design and develop our system in a Web-based manner. Therefore, we needed to provide a Web-based flowchart diagram user interface for an intuitive learning design user experience, and the available Web-based flowchart diagram technology was not so mature as the available desktop-based diagram technology.

Importing External Service Britain (2004) also mentioned that many more collaborative and other learning services were needed for Learning Design. The question was how to support these in an open and interoperable way. The required services included email service, conferencing service, multimedia content service, etc. Associated with the service importing, the access permissions, such as read-only, read and write, etc., also needed be specified for different roles, such as participant, group leader, observer, facilitator, administrator, etc.

Besides, there were also some other requirements, such as manipulating the three-level structure of PBL scripts, having restriction avoiding invalid authoring, providing context-aware authoring functions, etc.

5.1.2 A PBL Runtime Tool

Only providing a tool for PBL script design was obviously not enough. If a PBL script was designed only for formal description, the facilitation for PBL would

be very limited; the benefits were restricted to the design stage. In other words, teachers also needed a tool to run PBL scripts and that the facilitation of PBL could further step into the implementation stage. In fact, a computer supported PBL implementation was also essential, since (1) we had elaborated that one major difficulty for PBL is that the teaching-learning activities of PBL was relatively more complicated than other or ordinary learning strategies. Without computer supported implementation, there could be no real sense of innovative improvement of PBL facilitation; (2) for prompting PBL, there was another objective in that we needed to provide a way to enable PBL teachers to improve their PBL practices more easily. However, without computer supported implementation, there was no efficient way of helping the evaluation and improvement iteration. In summary, a visual authoring tool and a PBL runtime needed to be considered as a minimal complementary tool set to accomplish the goal of supporting technology-enhanced PBL.

Of course, according to the introduction in Section 3.3, there was an IMS-LD engine. Since the PBL scripting language could be seen as a sub-set of IMS-LD, it was possible to make PBL scripts be interpreted by computers in this way: (1) design and develop a transformation function to translate PBL scripts to IMS-LD *units of learning*; (2) use the IMS-LD engine to interpret *units of learning*; and 3) make use of IMS-LD compliant learning interaction user interfaces to let PBL scripts run in front of learners. Figure 5.2 depicts this way.

Nevertheless, there was a significant lack of solution, that is, currently no mature IMS-LD compliant learning interaction user interface existed. The state of art review in Section 3.3 showed this situation. Because of this, we had to develop our own native *PBL Runtime Tool*.

As Figure 5.3 illustrates, similar to the runtime constitution of IMS-LD, our *PBL Runtime Tool* needed to consist of two major components: a PBL runtime engine and a PBL runtime user interface. The greatest benefit of this decoupling was that the PBL runtime engine could also be exposed as an application public interface (API) servicing third-party software. Other organizations would also be able to develop their own PBL runtime user interface either for the reason of their special needs or providing implementation or any others, etc. Besides, it was still meaningful to develop a transformation function to map PBL scripts to *units of learning*. The significance was that we could leverage the IMS-LD ecosystem because there at least already been a powerful engine that supported all three levels

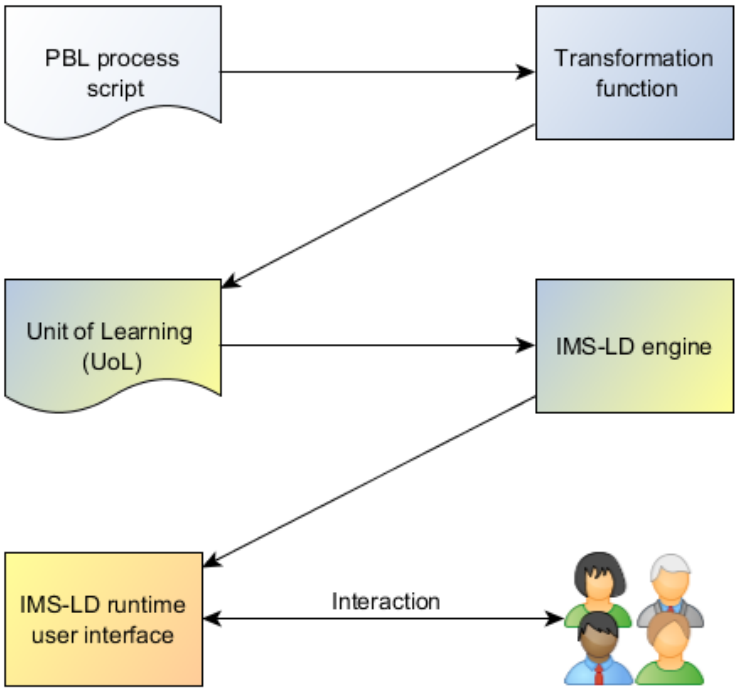


Figure 5.2: The way of making PBL scripts runnable in IMS-LD context for providing learning interaction.

of IMS-LD (Level A, Level B, and Level C. See Section 2.3.3).

5.1.3 System Architecture Design as the Core Challenge

To develop a visual authoring tool or to develop a runtime tool, there was a precondition that the PBL scripting language needed to be well defined. However, when PLATE project was established, the PBL scripting language was more or less just a basic idea. Although it was seen as the cornerstone of the PLATE system, it was far from being a well defined mature scripting language. In fact, according to the plan of the PLATE project, develop a PBL scripting language was task. In the proposal of the PLATE, the original sentence was stated as the following:

> Developing a PBL scripting language to enable the representation of a wide range of PBL models, and investigating the expressiveness of that language.

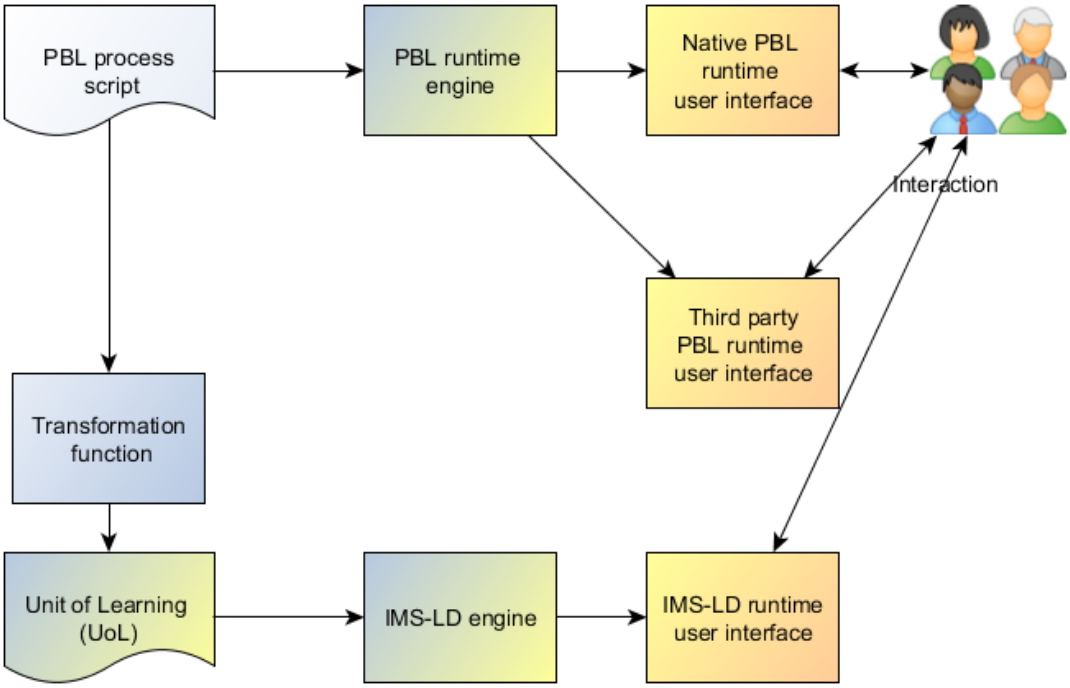


Figure 5.3: Developing a PBL specific engine and a native runtime tool was necessary.

So, what does this situation mean? It meant that **we needed to develop a system which would be built upon a fundamental specification, where the specification would be developed and investigated through the system itself**. This further meant that the system development of the PLATE project have to face a large scale of **mutual recursion** problem.

In Section 4.1, I stated that there was a considerable risk that if there was any imperfection of the standard, for example, some concepts needed to be deprecated or added, or even the concept structure needed to be redesigned and modified or any other revisions needed, then all the technical implementations based on the standard would encounter fundamental problems or even failure. However, now we were experiencing this risk.

Challenges of developing the visual authoring tool and the runtime tool were relatively easy to overcome since they were low-level function design and implementation problems. The challenge of solving the large scale mutual recursion problems of the system, however, was an entirely different engineering problem that needed to be overcome, because it was a high level system architecture design and implementation problem. Especially, the design and development of the visual author-

ing tool and the runtime tool relied on the architectural challenge being overcome. Therefore, an advanced and sophisticated system architecture had to be carefully and thoughtfully designed. To overcome this challenge was considered as one of the most crucial tasks in order to accomplish our project plan.

5.2 Learning Design as Model-driven Development

Although Miao et al. (2015b) proposed to develop a PBL scripting language, that could be referred to as a fundamental idea to support the development work of the PLATE project, this idea was more or less a conceptual guidance, which was far less than enough for the technical design and implementation. Since the PBL scripting language was a kind of domain specific modeling language and domain-specific modeling (DSM) as a software engineering methodology for designing and developing systems was developed mainly for industrial production purposes for a long time and tends to be mature, this chapter will center around the concept of modeling, analyzing the industrial model-driven approach, and then mapping the approach to our problem-based learning design context in order to gain a concrete technical design and implementation methodology for the PLATE project.

According to Mellor et al. (2003), if we followed a notion that we intended to construct a model of a system that we could then transform into the real thing, it simply meant that we were performing the model-driven development (MDD or MDE)[1]. Mapping this sentence to our context, (1) the activity of constructing a model was the activity of PBL Learning Design, and 2) transforming the system model to a real thing indicated interpreting the PBL model to an interoperable runtime software. Therefore, from the perspective of software engineering, facilitating teachers to act the PBL design was similar to supporting developers to perform the model-driven development.

As the finding stated above, to some extent, Learning Design was similar to model-driven development. Furthermore, in order to more precisely reference model-driven development, we needed to identify the specific similarities. Doing so pro-

[1]For this section that, MDD is a software development methodology in software engineering. Therefore, from an engineering point of view, people also call model-driven development as model-driven engineering (MDE). In order words, in this thesis, these two terms, MDD and MDE, are treated interchangeably.

vided important guidance for our system design and development.

5.2.1 With Same Intention

Similar to Learning Design, there was no universally accepted precise definition of model-driven development. However, people knew why we needed it, how to work it out, and what goal needed to be potentially achieved. In "Model-driven development: a metamodeling foundation," Atkinson and Kuhne (2003) stated that:

> "Ever since human beings started using computers, researches have been working to raise the abstraction level at which software engineers write programs. The first Fortran compiler was a major milestone in computer science because, for the first time, it let programmers specify what the machine should do rather than how it should do it. Since then, engineers have continued apace to raise programming abstraction levels. Today's object-oriented languages let programmers tackle problems of a complexity they never dreamed of in the early days of programming. Model-driven development is a natural continuation of this trend. Instead of requiring developers to spell out every detail of a system's implementation using a programming language, it lets them model what functionality is needed and what overall architecture the system should have. Nowadays, compilers automatically handle issues such as object allocation, method lookup, and exception handling, which were programmed manually just a few years ago."

From this statement, we found that there was almost the exact same situation in the current Learning Design. We could simply replace a few words in this statement to re-state it as the follows: Ever since educational practitioners started using computers to enhance and facilitate activity-oriented learning, pedagogical researchers have worked to provide a high abstraction level at which teachers design learning processes. The first educational modeling language, EML, was a major milestone in computer supported Learning Design; for the first time, it let teachers specify what learning tools should do for students rather than tell students how to use different learning tools. Since then, pedagogical researchers continued apace to refine abstraction description. Later, IMS Learning Design let educational practitioners achieve the computer supported learning design of

flexibility and expressiveness they never thought of in the early days. Domain-specific Learning Design such as the PBL scripting language based Learning Design was a natural continuation of this trend. Instead of requiring teachers to use general concepts for describing PBL processes, it let them model what special functionality was needed and what overall activity pattern learning should be. Currently, existing Learning Design runtime engines automatically handled issues such as learning content allocation, activity lookup, and condition handling, which were only programmed manually just a few years ago.

Furthermore, as the features that could be borrowed to guide Learning Design, Atkinson and Kuhne (2003) stated that "MDD aims to automate many of the complex (but routine) programming tasks—such as providing support for system persistence, interoperability, and distribution—which still have to be done manually today;" Mellor et al. (2003) stated that "Model-driven development captures expert knowledge as mapping functions that transform between one model and another. Executing those mapping functions transforms one model into another form" and "Mapping functions capture expert knowledge, so designers can reuse it when an application changes or when any of the technologies the application desuspends on change;" and Selic (2003) said that "Models help us understand a complex problem and its potential solutions through abstraction." From these statements we could summarize the following:

- MDD aimed at using simple concepts to represent the complex routine task. Learning Design using the PBL scripting language had the same objective. For example, if we annotated a learning activity *Brainstorming* with an attribute *Collaboratively*, it meant there would be a learning scenario that several students could collaboratively perform brainstorming through. Concepts in this description were very simple while there were many of complex work that needed be done by runtime tools, such as start brainstorming service, load user information into service, render use interface, handle different input from different user, forward different messages to different user, data persistence or retrieval, users access control, and other complex work.
- MDD aimed to let experts take the responsibility of overcoming the difficulty of providing the simple concepts and transforming simple concepts to complex runtime executable activities. Learning Design using the PBL scripting language aslo had the same aim. PBL pedagogy experts needed to review many different learning models to extract high level, but easy to understand, concepts, such as *Problem definition*, *Identification of learning issue*, etc. and

define the transformation method. For example, *Problem definition* would let runtime tools provide students a user interface with some redefined read only information. *Identification of learning issue* would let runtime tools provide each student a user interface with some writable input text areas.

5.2.2 With Same Working Process

Not long ago, when developing a software, there was only one way: using machine-oriented programming language such as Java, which could be considered as a very low abstract level modeling language, to write source code implying to model a system, then runtime platforms such as Java Runtime Environment (JRE), executed the source code to enable the system model to have behavior. Nowadays, with the emerging of higher level modeling technology, model-driven development dramatically changed the way we develop applications. Figure 5.4 depicts a typical scenario of the model-driven development for the software engineering in industry (Kent, 2002; Selic, 2003; Aßmann et al., 2006; Schmidt, 2006). From this scenario, we could see that system developers no longer needed to directly write source code, instead, they could use some high abstract level modeling language, such as Unified Modeling Language (UML), with the help of visual modeling tools to model systems (the benefits were introduced in the last section). Then the visual modeling tools took the responsibility to transform the human-oriented high abstract level models into machine-oriented low abstract level models such as Java source code. Then runtime platforms, such as JRE executed the source code to make the high abstract level system models have behavior. In other words, software development did not need to focus on the coding details but advanced models. When there was something in the runtime that needed to be improved or modified, system developers simply improved or modified the high abstract level system models, rather than improve or modify the source code.

In Section 2.3.1, Learning Design was described as a way that teachers use a formal description to structure the sequence of learning activities in order to reflect the learning need of students, where then the structured sequences of information and activities could be carried out via computers, together with the necessary supports from teachers, to promote technology enhanced learning. According to this description, we could have the following (PBL specific) Learning Design scenario shown by Figure 5.5.

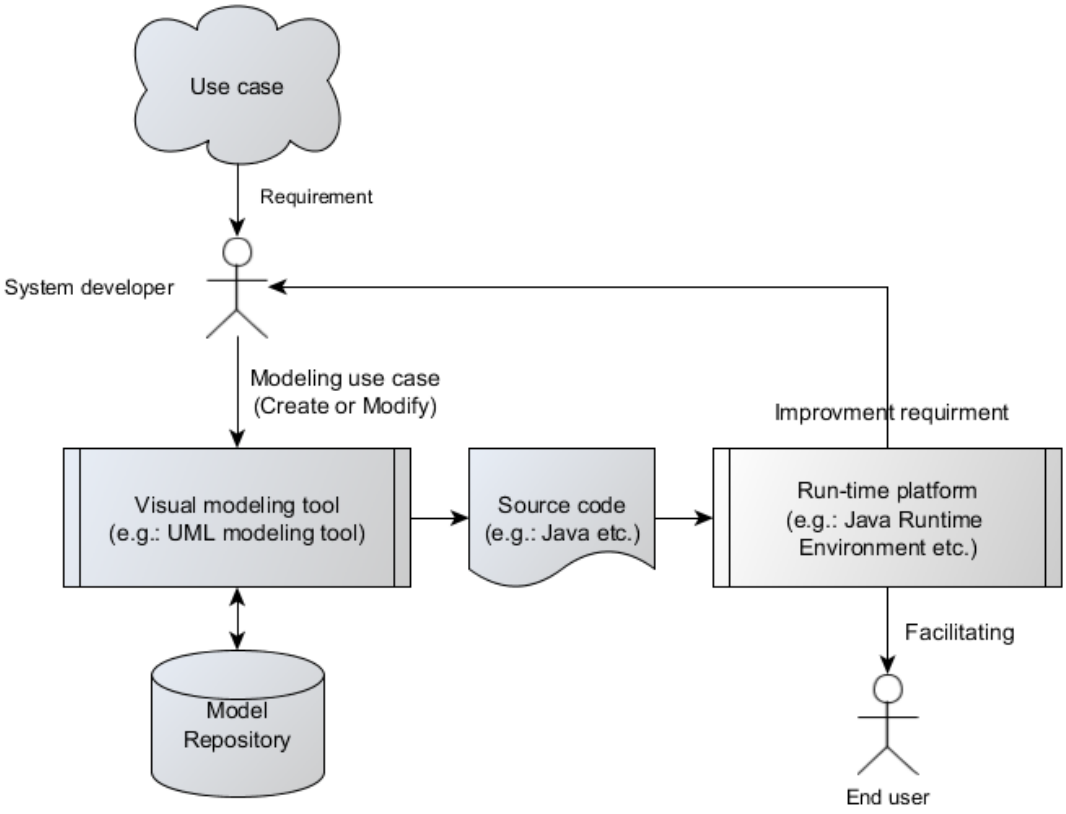


Figure 5.4: A typical scenario of the model-driven development in software engineering.

In (PBL specific) Learning Design, designers (teachers) found learning needs from learning scenarios, then used a high-level modeling language, such as the IMS-LD modeling language (or the PBL scripting language), with the help of virtual learning design modeling tools to model the learning processes. The modeling tools would generate computational learning process scripts. At last, computer-supported learning tools as the runtime platforms to execute the learning process scripts in order to made learning process models have real behavior. When there was a need to improve or modify learning processes or contents for runtime, designers (teachers) did not redevelop the runtime tools but improved or modified the learning process models.

5.2.3 Summary

From the comparison above, we could summarize that, (PBL specific) Learning Design had the similar intention with model-driving development. The similarities

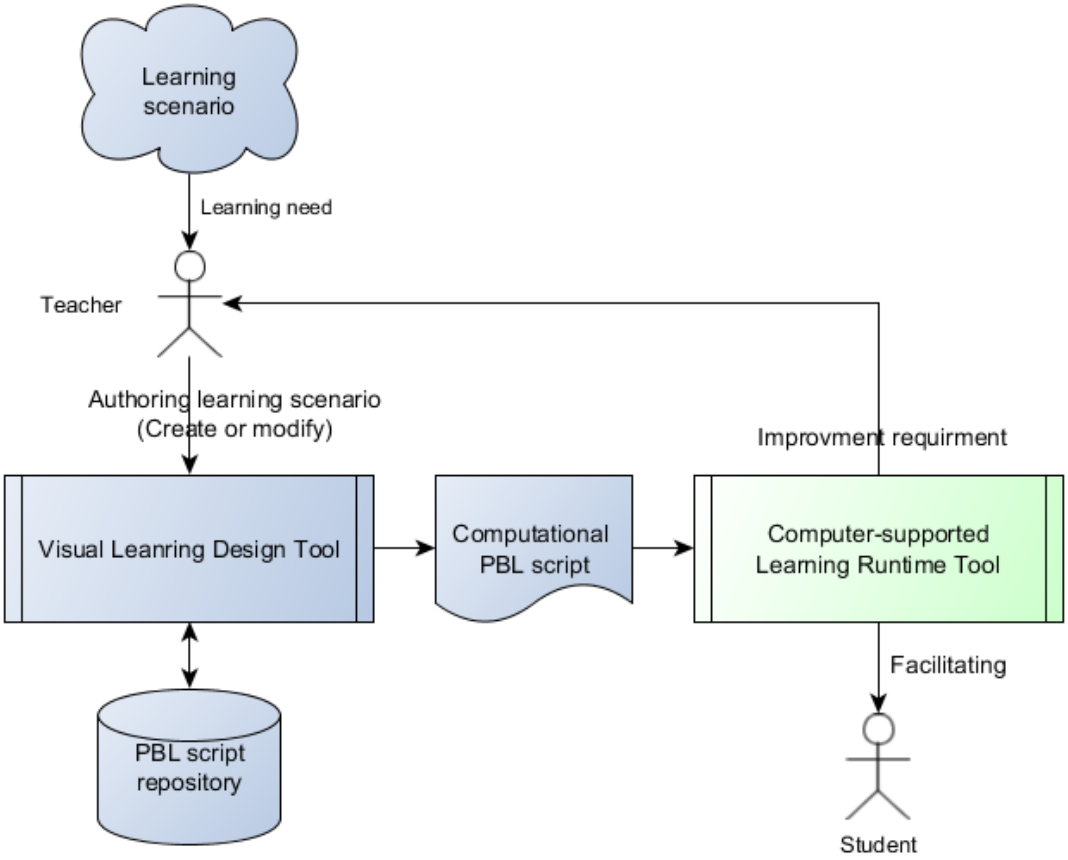


Figure 5.5: A typical scenario of the Learning Design for technology enhanced learning.

included:

- **Similar motivation**: Learning Design or the system development was complex; it is difficult for learning designers or system developers to hold, etc., Therefore we needed a way to provide more productivity.
- **Similar major work**: Define modeling language, specify transformation methods to let the model become a runnable instance, etc.
- **Similar target goal** Facilitate Learning Design or system development, reduce the communication difficulty, increase reusability, etc.

On the other hand, (PBL specific) Learning Design also had the same working process as the model-driven development. The similarities included:

- **Similar problem source**: Learning scenario was a kind of use case.
- **Similar modeler**: Teacher was the designer and also a kind of developer.
- **Using similar tool**: Virtual modeling tool.

- **Run in similar way**: Generated computational processable data and a runtime platform executed the data.
- **Similar improvement or modification**: Improve or modify model to affect runtime behavior.
- **Similar target user**: Students were a system end user.

Therefore, we could clearly conclude that (PBL specific) Learning Design was essentially identical to the model-driven development. We could even generally say that (PBL specific) Learning Design was a kind of model-driven development. Therefore, we needed to reference this mature methodology from model-driven development as much as possible in order to guide the technical design and implementation work of the PLATE project. Now we had to answer: how was model-driven development implemented in software engineering.

5.3 Infrastructure for MDD: Model-driven Architecture (MDA)

To support the MDD in software engineering, there was extremely important specification infrastructure, which provided a set of guidelines, named Model-driven Architecture (MDA).

Shaw and Garlan (1996) pointed out that "MDA is an approach to system development, which increases the power of models in that work. It is model-driven because it provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification". additionally, they said that this architecture "provided a specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors". According to OMG (2003) and Mellor et al. (2003), MDA was a set of OMG (Object Management Group[2]) standards that enabled the specification of models and their transformation into other models or systems. MDA separated subject matters so that application-oriented models were indesuspendently reusable across multiple implementations and vice versa.

[2]Object Management Group is an international, not-for-profit industrial consortium that creates and maintains software interoperability specifications. The OMG's specifications include the UML modeling notation, XMI (XML metadata interchange), CORBA (common object request broker architecture) middleware, and dozens of domain-specific interoperability specifications in such areas as transportation, life sciences, telecommunications, and manufacturing (Mellor et al., 2003).

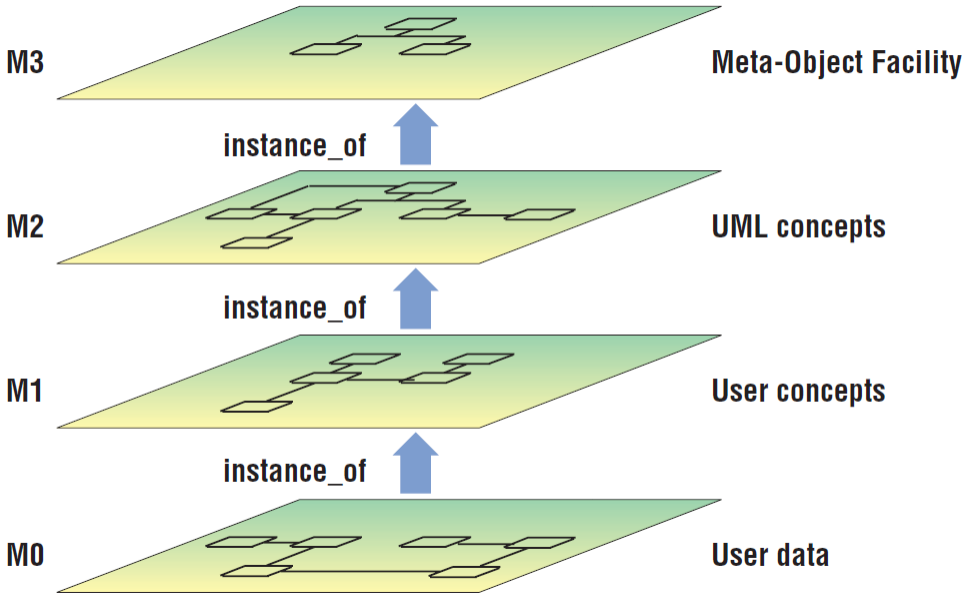


Figure 5.6: MDA: A framework consisting of a four-layer hierarchical structure (Atkinson and Kuhne, 2003).

The kernel idea of MDA was a layered framework (Kleppe et al., 2003). Figure 5.6, borrowed from (Atkinson and Kuhne, 2003), shows this hierarchy. MDA told us that a model-driven system could be built on models of different abstract levels from different viewpoints. These different levels of models were organized based on a relationship of hierarchy, where models of each level were the instances of the models of the next upper level. Meaning, in M0 layer the models were the final computer manipulable objects containing user data for user interaction. Those objects were the instances of the models in M1 layer. M1 layer models were the conceptual models. They could not directly be executed by computers for user interaction but could be used to represent the patterns of human-computer interaction. The conceptual models were the instance of the M2 layer models. M2 layer models defined the concept set relationships between the concepts. They were modeling languages. In the context of MDA, the Unified Modeling Language (UML) was an M2 layer model. Languages could be still the instances of another special model. An M3 model was type system for entities and a set of interfaces through which those types could be created and manipulated for entities. In MDA context, the M3 model was the MetaObject Facility[3] (MOF).

[3]The MetaObject Facility Specification is the foundation of OMG's industry-standard environment where models can be exported from one application, imported into another, transported across a network, stored in a repository and then retrieved, rendered into different formats (in-

In this modeling division, the M1 indicated the **models** that we usually talked about in default. For example, *units of learning* or PBL process models (the models we were going to provide support to design) were all M1 models, namely, platform specific models. Based on this naming convention, we could then call M2 as **metamodel**, M3 as **metametamodel**, and M0 as concrete **instance**.

5.4 Mapping Concepts of MDA to PBL

Now we had to answer, the following questions: how do we apply the PBL scripting language to MDA? Was the PBL scripting language suitable to be used in MDD? Since the PBL scripting language, as the name suggests, was a language, we should analyze it first from a computer linguistics point of view?

5.4.1 Linguistic Analysis

According to the specification of MDA, the PBL scripting language should be located in layer M2. Figure 5.7 shows that an activity concept *Identity learning issue* was used as an example to illustrate this linguistic analysis of the PBL scripting language.

In this figure, the learning scenario *Identify learning issue* was the target which would be finally manipulated by computers for the learning interaction. The learning scenario needed to be placed in the layer M0. Since it was the target, then everything used to describe it needed also be abstracted from it. There were two things to be extracted. The first was the construction principles of the scenario; the second was the behavior characteristics of the scenario. In these, the abstraction of the construction principles was the work of PBL pedagogy experts, while the representation of the behavior characteristics was the work of PBL teachers. The goal of the PLATE project was to provide support for PBL teachers to let them easily represent the learning activity behavior.

The first level result of the abstraction of the construction principles was to pro-

cluding XMI, OMG's XML-based standard format for model transmission and storage), transformed, and used to generate application code. These functions are not restricted to structural models, or even to models defined in UML - behavioral models and data models also participate in this environment, and non-UML modeling languages can partake as well, as long as they are MOF-based. Quoted from http://www.omg.org/mof

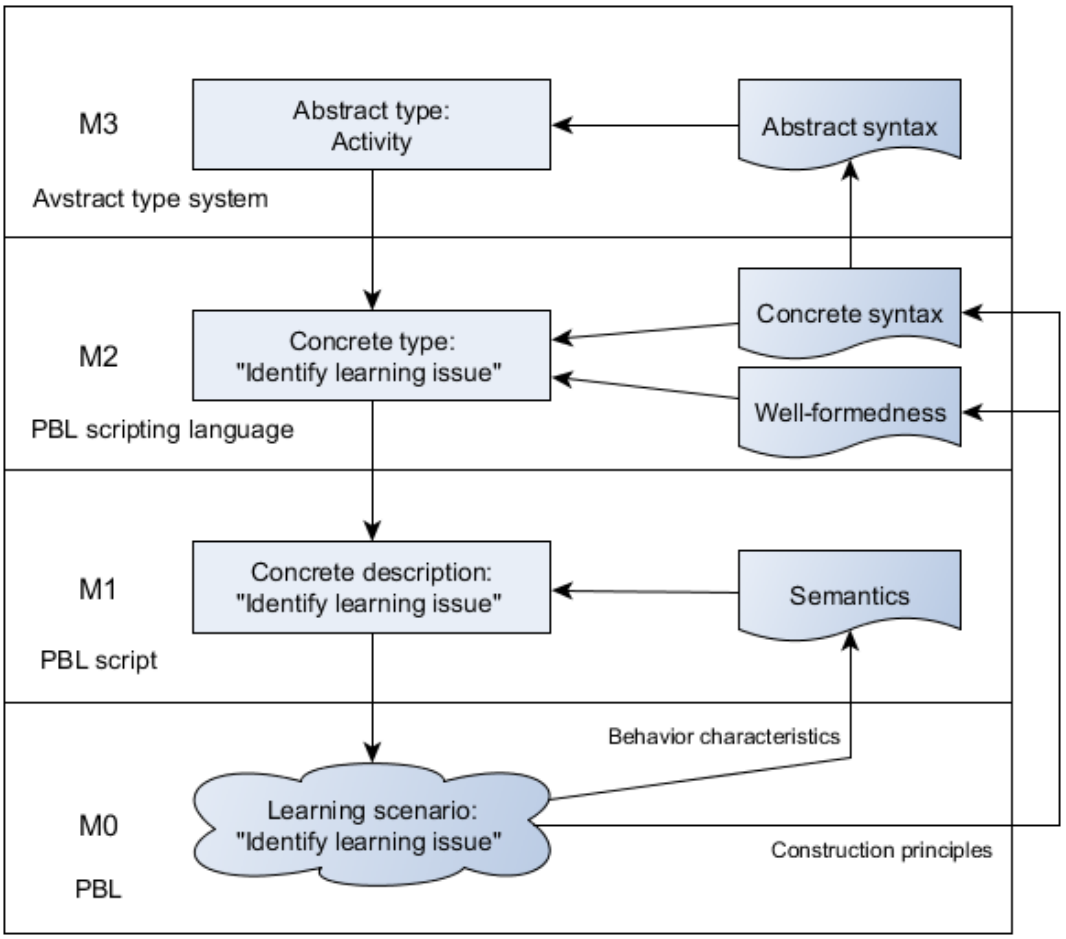


Figure 5.7: A linguistic analysis of the PBL scripting language.

duce a concrete syntax and a well-formedness. Concrete syntax was partially introduced in Section 4.4 and was the notation used in depicting models. For example, a concrete syntax could be a name description, function description, pre-conditions, expected outcome, etc. of the scenario *Identify learning issue* while well-formedness indicated the rules for applying the notation. For example, the scenario *Identify learning issue* was not allowed to be started without any participant. Based on the concrete syntax and the well-formedness, teachers was able to author their concrete scenario descriptions.

By now we had located the PBL scripting language to layer M2 and its another two instances to layer M1 and M0. Next, we needed to identify the M3 layer in MDA for the PBL scripting language. In fact, similar to the conceptual structure of the IMS-LD specification (see Figure 2.3), we could and needed be also have a higher but clear abstraction of the language. People also referred to this kind

of abstraction as metalanguage[4]. The abstraction of the PBL scripting language consisted of an abstract syntax. Concepts of the abstract syntax indicated the PBL scripting language from which ontology types and rules were created. As an example, in Figure 5.7, the abstract description of the *Identify learning issue* was *activity*. This meant a *Identify learning issue* must be a *activity*, while an *activity* could represent the things that people did, especially in order to achieve a particular aim, which did not mean to perform the *Identify learning issue*.

5.4.2 Concepts

According to MDA Guide Version 1.0.1 (OMG, 2003), MDA Guide Version 2.0 (OMG, 2014), Kleppe et al. (2003), Stahl et al. (2006), and Liddle (2011), there were another group of important concepts in the MDA, which included Platform Independent Model (PIM), Platform Specific Model (PSM), and Model Transformation. Those concepts made MDA work. This section will present an investigation whether the PBL scripting language based Learning Design can be mapped to those concepts or not. If yes, we needed to specify how to complete the mapping?

Platform Independent Model (PIM) was a view of a system from the platform independent viewpoint. A PIM exhibited a sufficient degree of independence to enable its mapping to one or more platforms. This was commonly achieved by defining a set of services in a way that abstracted technical details. Other models then specified a realization of these services in a platform specific manner.

According to the definition above, the PBL script is PBL-PIM. As discussed in Section 5.1.2, we needed to generate PBL scripts not only for the runtime purpose of the PLATE project itself but also for third-party usage. For example, it was significant to leverage the IMS-LD ecosystem, to have the IMS-LD runtime engine and the runtime user interface be compliant to our PBL scripts.

Platform Specific Model (PSM) PSM was a view of a system from the platform specific viewpoint. A PSM combined the specifications in the PIM with the details that specified how that system used a particular type of platform.

According to this definition, a PBL script could also be the PBL-PSM when

[4]Metalanguage: A form of language or set of terms used for the description or analysis of another language (Definition from the Internet).

it was ready for use in specific runtime engines. In this form, a PBL script contained the necessary "dialect" details. For example, PBL-PSM could be a dialect either for the PLATE project native engine or IMS-LD engine.

Model Transformation was the process of converting one model to another model of the same system.

By using the analyzed mapping from above for the concept platform independent model and platform specific model, there was one important transformation for the PLATE project - transform PBL-PIM to PBL-PSM. The general pattern was:

$$Target\ model = Source\ model + Transformation\ rules$$

Figure 5.8 shows the concept mapping from MDA to PBL model-driven development through using a concrete example "Deformed Frogs." which was introduced in Section 4.4.1. The last important mapping was when we mapped PBL model-driven development in MDA:

- the syntax of the PBL scripting language mentioned in the last section corresponded with the metametamodel; the PBL scripting language corresponded with the metamodel;
- PBL-PIM, and PBL-PSM were all part of the PBL model. The difference was in the viewpoint. PBL-PIM was for multiple runtime transformation and model storage use, while PBL-PSM was the transformation result of PBL-PIM;
- a PBL script running in PBL engine was referred to as a PBL instance, where an instance would hold the user data such as specific user information, adaptive learning contents, particular start condition of a certain learning activity, newly generated information, etc.

5.5 Iterative Development of the PBL Scripting Language

The above elaborated that we could apply the methodology of a model-driven development on the PBL scripting language to build the problem-based learning design and runtime system. This feasibility of this approach was based on an

- v3.png

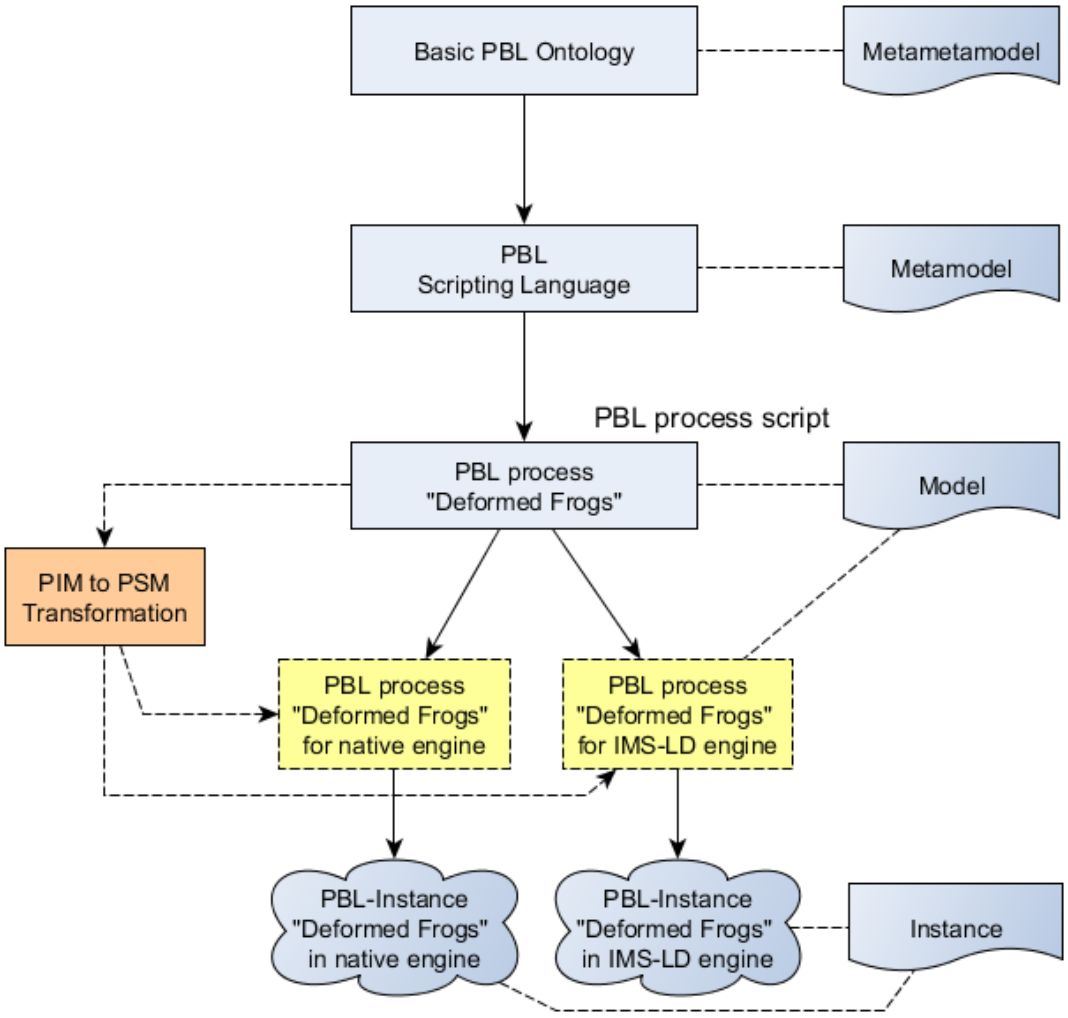


Figure 5.8: Mapping concepts of MDA to the PBL model-driven development.

assumption that the PBL scripting language, as a DSL, was well defined. However, this was not true. Currently, we only had an initial idea of the scripting language, which aimed to specifically support technology enhanced PBL. In fact, as it was mentioned in the requirement analysis section, one of our system development tasks was to develop the scripting language in order to enable the representation of a wide range of PBL models. Moreover, the investigation of the expressiveness of the language was also the work included in the development task. Therefore, the next questions: how to integrate the development of the PBL scripting language into the model-driven problem-based Learning Design and how to overcome the large-scale **mutual recursion** problem.

5.5.1 Waterfall Model was not Suitable

In "DSL engineering: Designing, implementing and using domain-specific languages," Voelter et al. (2013) figured out that there was a "crucial challenge in DSL design: finding regularity in a non-regular domain and capturing it in a language. Especially in the deductive approach, membership of programs in the domain is determined by a human and is, in some sense, arbitrary." This was fully proved. In the first half of the PBL project time, there was a plenty of time that was spent in determining problems, such as what the name of an activity should be able more appropriately closer to teachers' practice, what activities should belong to which phase, what the common activities for different phases were, what the properties of an activity or a phase were, which constrains needed to be provided, etc. Voelter et al. (2013) continued, "Some people use DSLs as an excuse to reintroduce waterfall[5] processes. They spend months and months developing languages, tools, and frameworks. Needless to say, this is not a very successful approach."

These statements were very enlightening for our system development. If we followed the waterfall model, as Figure 5.9 shows, we needed to first finalize the specification of the PBL scripting language before we started the implementation work. In this way, our system would not allow much reflection or revision for the PBL scripting language. The problem that remained was if we found any insufficiency of the language in the stage of supporting Learning Design and runtime, the stiff system implementation made it very difficult to change the language, such as a bottom foundation, of the system. Hence, we concluded that the waterfall was not suitable for our system requirements.

Voelter et al. (2013) also claimed that "a DSL for the domain hence typically represents an explanation or interpretation of the domain, and often requires tradeoffs by under- or over-approximation. This is particularly the case while we develop the DSL: an **iterative approach** is necessary that evolves the language as our understanding of the domain becomes more and more refined over time." Figure 5.10 shows this refinement process of the DSL development.

[5]The waterfall model indicates a systems development life cycle pattern. This model describes a development as is linear and sequential pattern by emphasizing there are distinct goals for each phase in software development. Like the waterfalls via steep cliff, water flows from one higher level to another lower level. However this also implies that the flowing process can not go backward, which means once a development phase is finished, the development work will go forward to the next phase. Development with turning back is not the right way (Boehm, 1988).

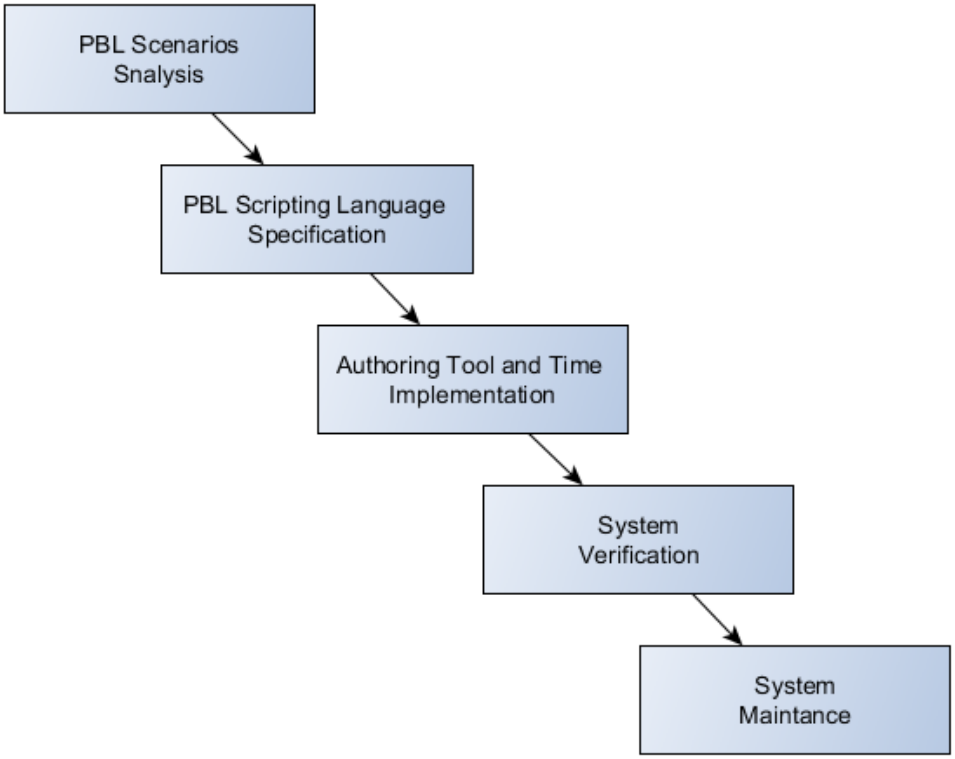


Figure 5.9: A waterfall model was not suited for the PBL script language development and the corresponding system development.

5.5.2 An Iterative Engineering Approach

Furthermore, Voelter et al. (2013) proposed that

> we need to iterate when developing the language. Start by developing some deep understanding of a small part of the domain for which you build the DSL. Then build a little bit of language, build a little bit of generator and develop a small example model to verify what you just did. Ideally, implement all aspects of the language and processor for each new domain requirement before focusing on new requirements.

Developing the PBL scripting language was actually the development process of PBL ontology. From developing ontology point of view, Noy et al. (2001) also argued that ontology development was necessarily an iterative process. While developing the PBL scripting language, we needed a simultaneous way to test the language. The test included using the language to carry out the problem-based Learning Design and used runtime engine to interpret the Learning Design artifact. When we found any improvements needed, we would then go back to modify the

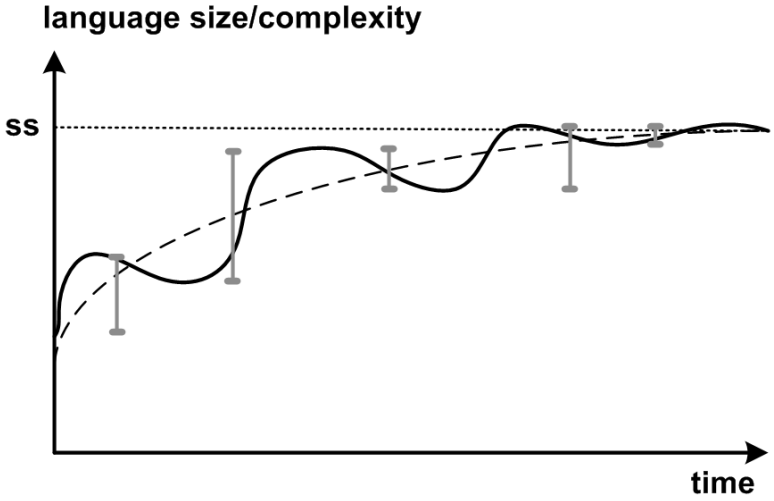


Figure 5.10: A refinement process of the DSL (or the PBL scripting language) development (Voelter et al., 2013).

language. This cycle could be seen as a development iteration. As a result, in each iteration, we improved not only the modeling language but also the authoring tool and the runtime engine of the Learning Design.

Since the PLATE project would both develop the PBL scripting language and the learning design authoring tool and runtime platform, people involved generally were divided into two user groups. The first was the PBL domain expert group. The domain experts were PBL pedagogy scholars who took the responsibility of the PBL domain analysis and the language definition. Of course, they could also use the language to author PBL scenarios. The other group was PBL domain user group. The domain users were PBL teachers and PBL students. Teachers used the PBL scripting language (via a virtual authoring tool) to author PBL scenarios. They would also later use a runtime tool, which was under the control of the designed scripts, to implement the PBL scenarios. PBL students were seen as the end users that only engaged in the PBL runtime activities.

Actually, by analyzing problem-based Learning Design as a kind of MDD in Section 5.2.2, we showed that the Learning Design could be done in an iterative manner. Through Figure 5.5, we illustrate how teachers, with the help of a visual learning design tool, produced a computational PBL script for computer-supported learning runtime platform use and how the runtime platform facilitated students in performing learning activities, which was equivalent to facilitating teachers in implementing PBL. This meant the domain users were placed in an iterative engineering context. Therefore, we could simply reuse this method for domain experts,

which would produce the similar activity diagram shown in Figure 5.11.

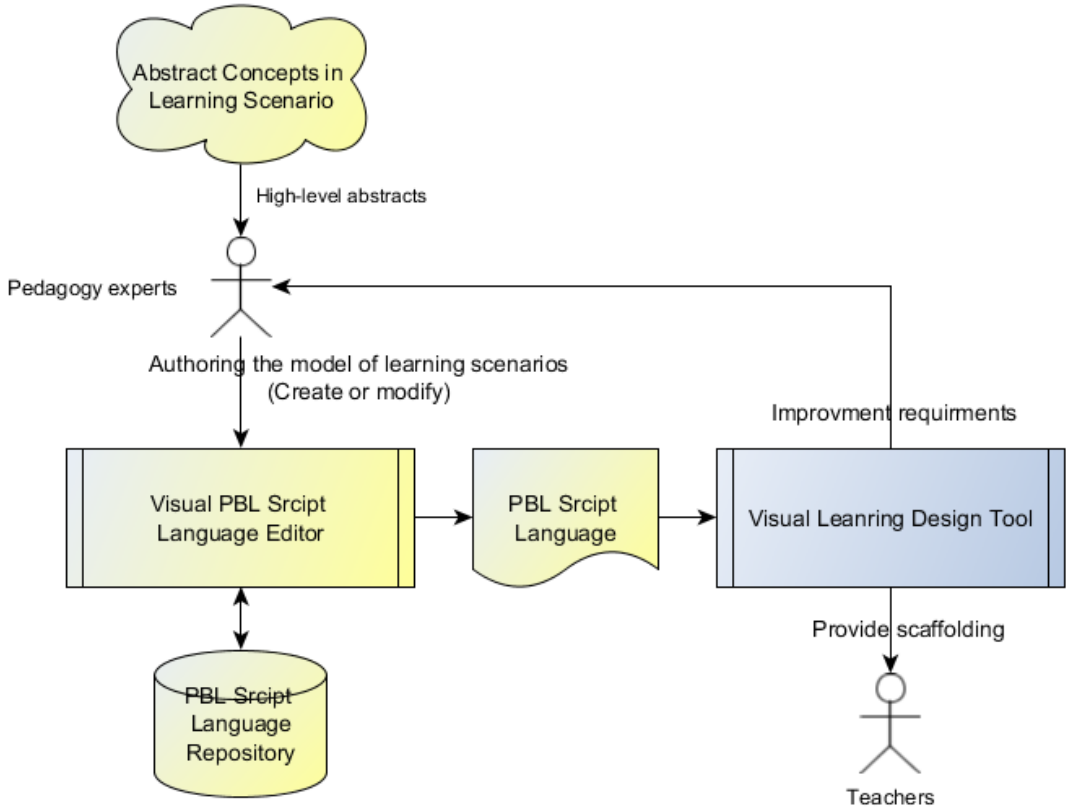


Figure 5.11: Inspired by Learning Design, an activity scenario of the PBL scripting language development.

Furthermore, if we combined the two iterative scenarios together, as shown in Figure 5.12, we could obtain our final approach to conducting the whole design and development approach for the PLATE project.

The start point of Figure 5.12 was the learning scenario. PBL Pedagogy experts or domain expert would first analyze different PBL patterns or frameworks, the PBL domain analysis. They extracted the common characteristics from different scenarios to form an abstract concept map in their mind. Then they used a visual PBL script language editor to formally represent the concept map. The outcome of the authoring was the PBL scripting language. The system could store different versions of the PBL scripting language. Experts could determine which version would be used for problem-based Learning Design. Each version was completed for the current observation, but it was still open for change and improvement. This mechanism could empower PBL experts to test their different considerations and findings.

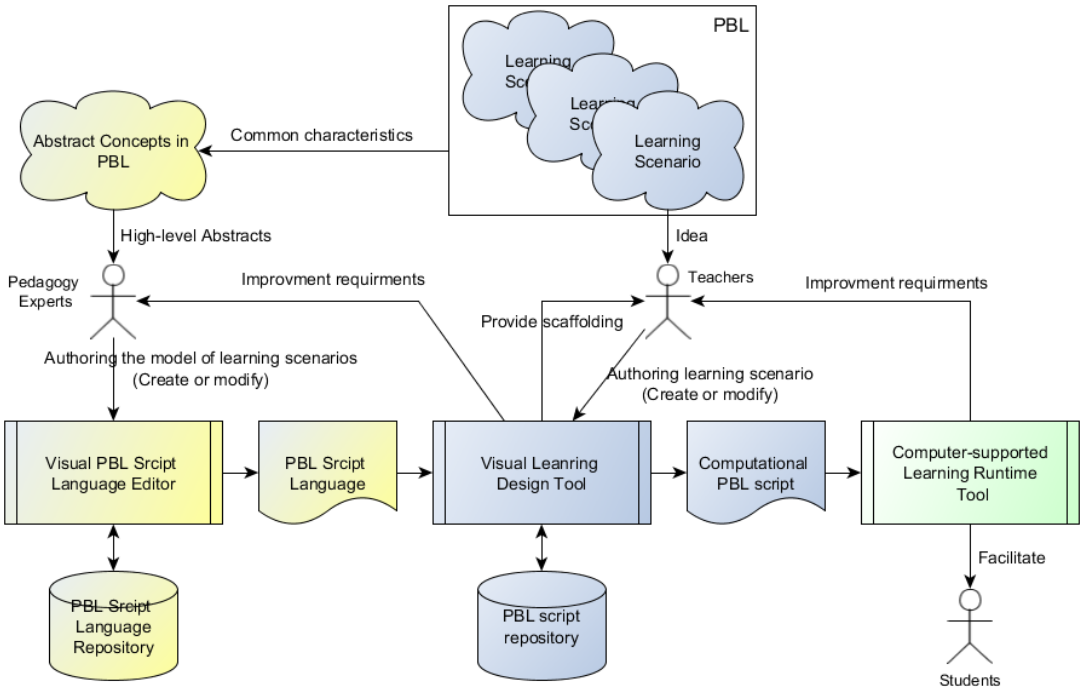


Figure 5.12: An iterative model-driven development concept in the entire system.

A visual Learning Design tool would then make the PBL scripting language easy to use. PBL teachers, or domain user, used the visual Learning Design tool to represent certain specific problem-based learning scenarios, and made their mantel feeling of the learning scenario to a formal computational PBL script. Teachers could also store any number of PBL scripts to represent different learning scenarios. A computer-support learning runtime tool would interpret the PBL script, and the runtime tool provided teach-learning content and interactions according to the teachers' plans, the PBL script, for students.

When teachers used the runtime tool to implement their PBL courses, they could find that something needed to be modified (e.g.: modify the process structure) or improved (e.g.: improve certain settings or control configurations) in order to improve a PBL course. The visual Learning Design tool could help them easily finish these above issues. However, sometimes a teacher could find that the visual Learning Design tool could not satisfy their modification, improvement, or representation of a certain scenario. This could mean that the PBL scripting language was still not mature enough. This feedback would be sent to PBL Pedagogy experts. This kind of feedback is usually very important for the improvement work for developing the PBL scripting language. After experts used the language ed-

itor to improve the language, teachers' previous modifications, improvements, or representation requirements would be possible to met. If there was something not good enough, either teachers or experts could re-operate the iterative improvement development cycle again and so on as necessary.

Finally, the goal of facilitating PBL implementation for teachers and providing innovative learning style for 21st century students would be achieved with a solid PBL pedagogical modeling research foundation.

Part II

Technical Implementation

Chapter 6

System Design

6.1 Basic Architecture

In the preface, it was introduced that my work was in the context of the project PLATE. In terms of the functionalities, the PLATE project required us to develop a high-level scripting language and a new generation of learning design environment that flexibly and efficiently supported the development of PBL infused modules and courses, as well as created a run-time environment that supports the execution of PBL models to scaffold and orchestrate PBL activities to help students transition into new roles and to learn within PBL contexts.

6.1.1 Constitution of PLATE

According to the goal of the whole project, the following objectives should be achieved. Those objectives made up a basic architecture of the system.

- **Develop a PBL scripting language** which represented a broad range of PBL practice frameworks.
- **Develop two PBL scripting language compatible authoring tools**, which included a form based authoring tool and a (graphical) flowchart based authoring tool, to support the use of the scripting language.
- **Develop two PBL runtime tools**. The first was an extension of Moodle; the other was ready for IMS-LD specification.
- **Integrate a repository** in the authoring environment to enable the storage, retrieval, comment, rate, and reuse of PBL scripts.

- **Develop PBL communities** to support the research and application of technology-enhanced PBL methodologies.

Based on these functionality requirements, a basic architecture of the system for PLATE could be depicted.

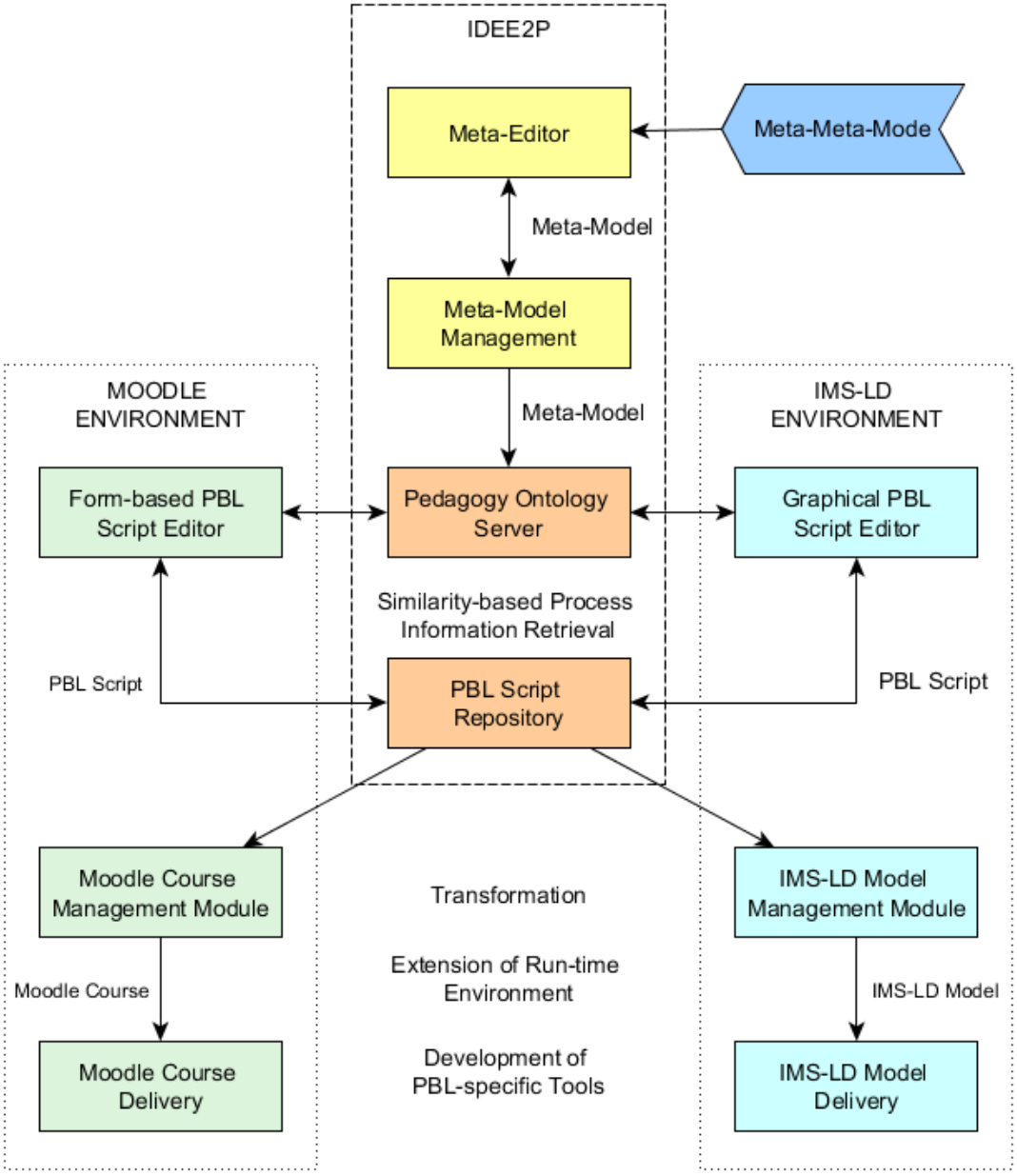


Figure 6.1: The conceptual architecture of the PLATE project.

Figure 6.1 illustrates those objectives. To achieve this, I was responsible for finding a way to help my colleagues to develop the PBL scripting language more easier, developing a flowchart based PBL scripting language compatible authoring tool

with a script repository, and developing a PBL runtime tool ready for the IMS-LD specification. These were my objectives. To achieve these, a PBL pedagogy analysis and model-driven engineering research was elaborated in detail in the previous sections. As a result, I conceptually concluded that it was necessary to use a model-driven engineering approach to realize these developments in an iteration and integration manner. Therefore I named the system under my responsibility **I**ntegrated **D**esign and **E**xecution **E**nvironment for **P**BL, shortly **IDEE4P**. To summarize, the IDEE4P needed to consist of the four major functional tools:

1. A visual PBL scripting language editor for PBL experts to carry out PBL metametamodel (language type system and abstract syntax) and metamodel (language element and concrete syntax) research.
2. A visual problem-based Learning Design tool providing an intuitive way to using the PBL scripting language for teachers (and experts) and to facilitate problem-based Learning Design (and to let experts test the language).
3. A native PBL script runtime instance management tool for both PBL teachers, PBL experts, and other educational stakeholders to arrange real participants to scripts, publish scripts as PBL courses, and perform other general PBL course management operations.
4. A native PBL course teaching-learning interaction user interface for PBL facilitator and students to carry out computer-supported learning activities. The runtime engine was underneath this tool.

6.1.2 Functional Architecture of IDEE4P

Technical Renaming

Before talking about the system technical architecture, it was necessary to emphasize a naming change problem. The reason was that in this thesis, Part I focused on the PBL pedagogy research and practice. Thus the words were given from the perspective of non-ICT-orientation. On the contrary, from this chapter, in Part II, since there was a need to elaborate the technical design and implement of the IDEE4P, I considered that it would be better to use a professional ICT engineering method to present the remaining chapters. The naming change is summarized as follows:

For the naming of tools:

- The visual PBL scripting language editor is named **PBL Language Editor**.
- The visual problem-based Learning Design tool is named **PBL Authoring Tool**.
- The native PBL script runtime instance management tool is named **PBL Course Management Tool**.
- The native PBL course teacher-learning interaction user interface is named **PBL Online Whiteboard**.

Correspondingly, the target user is renamed as the following:

- The user of *PBL Language Editor* is named **PBL Language Developer**, which generally will indicate PBL experts.
- The user of *PBL Authoring Tool* is named **PBL Designer**, which generally will indicate teachers for PBL education purposes and experts for PBL language research purposes.
- The user of *PBL Course Management Tool* is named **PBL Course Administrator**, which generally will indicates PBL teachers, PBL experts, or other PBL educational stakeholders.
- The user of *PBL Online Whiteboard* is named **PBL Participant**, which generally will indicate PBL facilitators and students.

It is noted that:

- PBL teachers or experts in the context of facilitating students in problem-based learning activities are called PBL facilitators.
- A PBL teacher can be both a PBL designer, PBL course administrator, and PBL participant.
- A PBL expert can be both a PBL scripting language developer, PBL designer, PBL course administrator, and PBL participant.

Basic Architectural

Figure 6.2 depicts the functional architecture of the IDEE4P consisting of these four tools summarized above through diagramming the major functional compositions of each tool, the data dependence relationship between the tools, the target users of each tool, and the way to provide computational service to external systems.

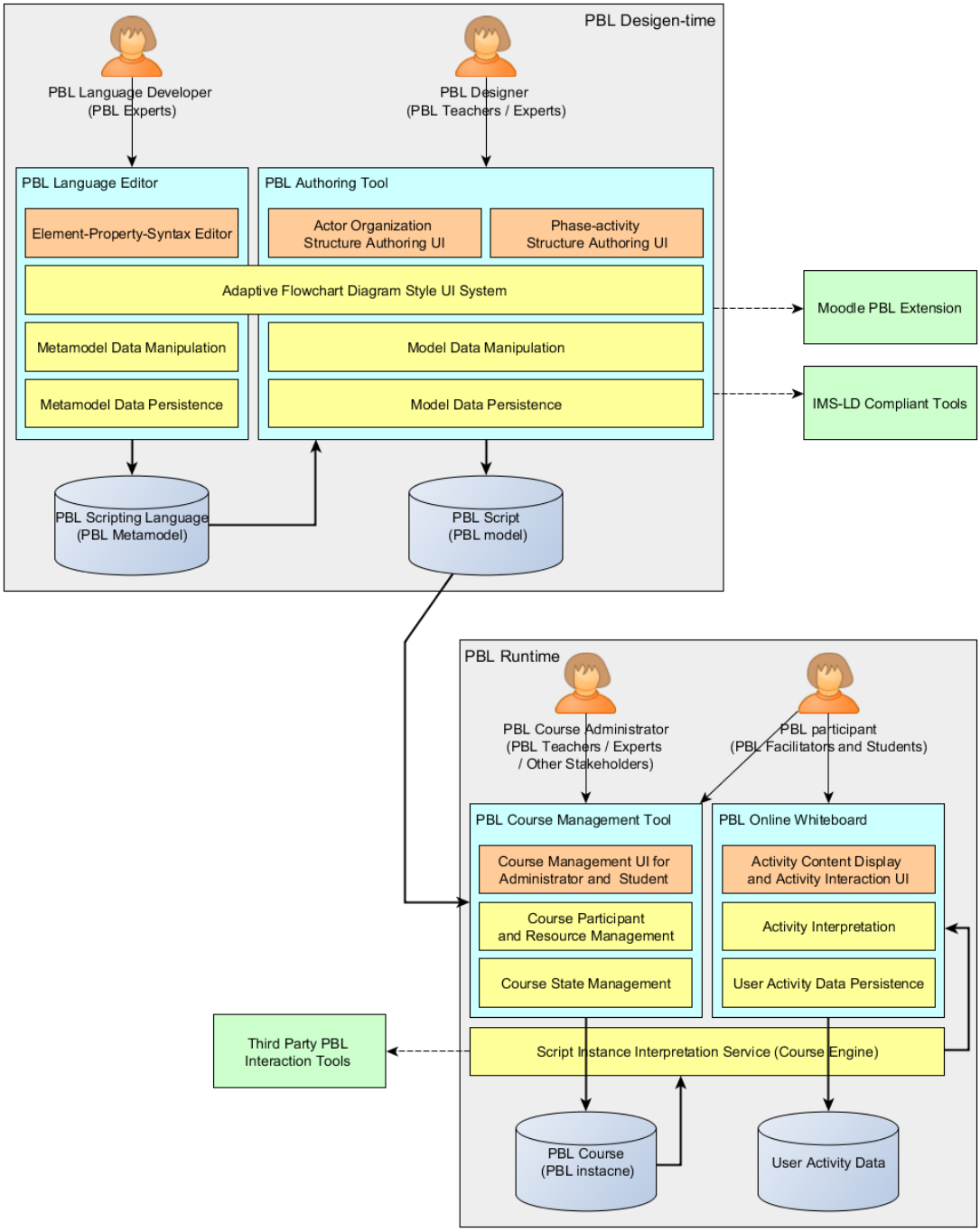


Figure 6.2: The functional architecture of IDEE4P (Integrated Design and Execution Environment for PBL)

As we can see from this diagram, the IDEE4P system was designed to have two parts: a **PBL Design-time** and a **PBL Runtime**. The *PBL Language Editor* and the *PBL Authoring Tool* belonged to the design-time; the *PBL Course Management Tool* and the *PBL Online Whiteboard* belonged to the runtime. In the

design-time, the authoring tool was driven by the PBL scripting language (PBL metamodel), whereas the PBL scripting language was the development outcome of the PBL experts. Supported by the authoring tool, a PBL designer could design the PBL process by authoring actor organizations and the phase-activity processes. As a result, a PBL script (PBL model) would be produced. Since the PBL scripting language was designed concerning the interoperability with IMS-LD, the PBL script was able to be transformed into *units of learning* and run in IMS-LD compatible tools. The compatible tool could be not only an activity "player", but also a Learning Design editor.

In the runtime, a PBL teacher (acts as PBL course administrator) would first use the *PBL Course Management Tool* to instantiate a PBL script into a PBL course through arranging participants, learning resources, and other settings such as start time. A script was then delivered as a PBL course (PBL instance) that was computational executable in the *PBL Online Whiteboard* and issued for students to take part in. Only involved participants (administrators, facilitators, and students) could see the issued courses. The *PBL Course Management Tool* would manage the status of the course (e.g.: a course was ready for choosing by students, or was running, or was finished, etc.).

Beyond these four tools, there was another special service: a *Script Instance Interpretation Service*, or *Course Runtime Engine*, was underneath the *PBL Course Management Tool* and the *PBL Online Whiteboard*. It was responsible for interpreting the script instances and understanding the semantics of learning processes based on the syntaxes defined in the scripting language and the scenarios described by PBL teachers. The engine interpreted the learning process, activity by activity, to assemble all the information, such as the profile information of participants, learning resources, third-party learning tools, expected learning artifacts of the current learning activity, etc., for the *PBL Online Whiteboard*. Then the whiteboard would disassemble the information and render them. Moreover, the whiteboard would also handle all the expected learning interaction of participants and store the data generated by participants in the learning processes.

To support those basic operation requirements, the top level function modules were organized as the following (the functional details of each module will be elaborated in Section 6.5):

PBL Language Editor consisted of a *Element-Property-Syntax Editor*, a *Metamodel Data Manipulation Module* , a *Metamodel Data Persistence Module*,

and a *PBL Scripting Language Database*.

PBL Authoring Tool consisted of an *Actor Organization Structure Authoring UI*, a *Phase-activity Structure Authoring UI*, a *Model Data Manipulation Module*, a *Model Data Persistence Module*, and a *PBL script Database*.

PBL Course Management Tool consisted of a *Course Administrator Management UI*, a *Course Student Management UI*, a *Course Participant and Resource Management Module*, a *Course State Management*, and a *PBL Course Database*.

PBL Online Whiteboard consisted of an *Activity Content Display and Activity Interaction UI*, a *Activity Interpretation Module*, a *User Activity Data Persistence Module*, and a *User Activity Data Database*.

In consideration for the common functional characteristics, there would be an *Adaptive Flowchart Diagram Style UI System Module* providing a workflow diagram style editing function for the *Element-Property-Syntax Editor*, *Actor Organization Structure Authoring UI*, and *Phase-activity Structure Authoring UI*. Additionally, there would be a *Script Instance Interpretation Service* providing the PBL instance status report and interpreting operation function for both the *PBL Course Management Tool* and *PBL Online Whiteboard*.

In consideration for providing service for external systems, the *PBL Authoring Tool* took upon the responsibility of providing the model transformation function to IMS-LD compliant tools and the Moodle PBL extension; the *Script Instance Interpretation Service* provided course management and interpretation services for third-party PBL interaction tools.

6.2 Ontology-based PBL Authoring

Figure 6.3 illustrates an ontological design following model-driven development (MDD) from the perspective of model transformation. It was noticed that, in order to simplify the presentation, this figure did not involve other instantiation managements such as learning resources arrangement etc.

In the IDEE4P, the PBL scripting language, as a domain-specific language (DSL), was the highest level model in the model-driven architecture (MDA). Therefore it was the initial point that drive the entire system. As mentioned above, the script-

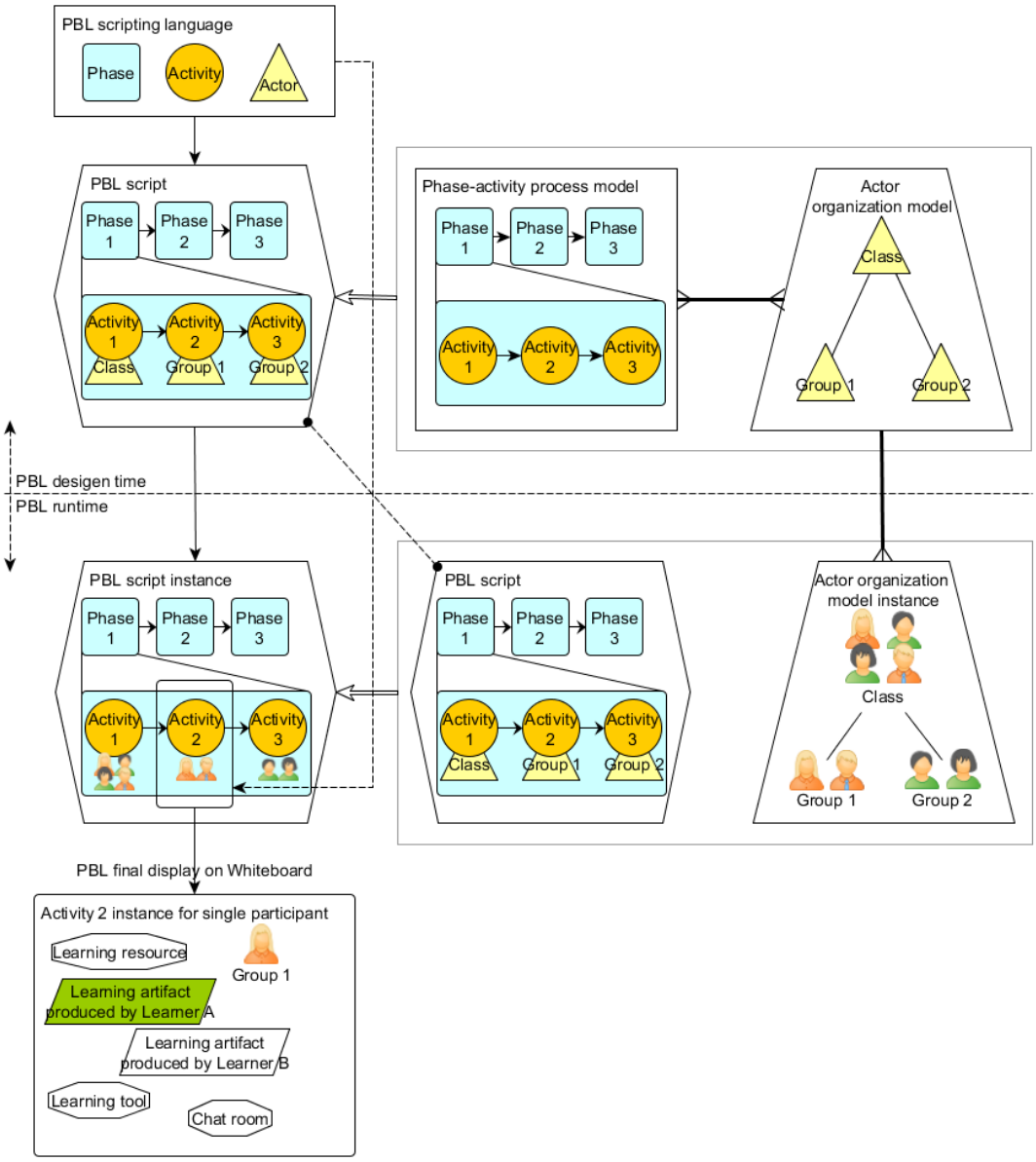


Figure 6.3: The model transformation in the IDEE4P system.

ing language was the metamodel of a problem-based learning process model. In the design time, the phase, activity, actor, etc. were the meta-elements, which were from the type system of the metamodel. These meta-elements were presented as the building blocks for designers to present the problem-based learning processes. With the help of the authoring tool, PBL designers made an ontological single learning process metamodel (PBL scripting language) become multiple different learning process models (PBL scripts). A learning process model consisted of a phase-activity process model and an actor organization model. The relationship

between these two models was *many-to-many*, which meant a phase-activity process could be alternatively combined with different actor organization models and vice versa.

At the run time, a learning process model (PBL script) could be transformed to multiple learning process model instances (runnable scripts) by arranging the different groups of real system users into the actor organization model (and by setting different learning resources, configuring different session start times, choosing different target run-time environments, etc.). The relationship between the organization model and the participant arrangement was *one-to-many*, which meant an organization model was able to have different arrangements. This enabled a PBL script to have infinite forms of instances. Finally, according to the specification of the learning process metamodel (the PBL scripting language), the possibility of interpreting every detailed interactive information of each activity in the learning process model instance was guaranteed.

As mentioned, the PBL script for representation purposes belonged to the platform-independent model (PIM) and the PBL script for external systems, such as the IMS-LD engine, belonged to the platform-specific model (PSM). There were already a number of tools that supported the PIM to PSM transformation, such as some computer-aided software engineering (CASE) tools. Nevertheless, they were mostly used for the code generation and designed for the technical people in the domain of software engineering, such as software developers, but not for the non-technical people such as PBL practitioners.

6.3 Runtime Logic

In this section, a course state transition diagram will be used to describe the basic principle of the runtime logic of a learning process (script) instance and how the runtime logic in a computer will be depicted.

6.3.1 Course State Transition

The script instance runtime logic management was another complex part of the IDEE4P system. To model this part of the process, I used the state-transition-diagram (STD) (Pressman, 2005). In this section, the state transition diagram

illustrates how the instance runtime logic behaved as a consequence where PBL course state transitions were made from one state to another state. Figure 6.4 shows this transition.

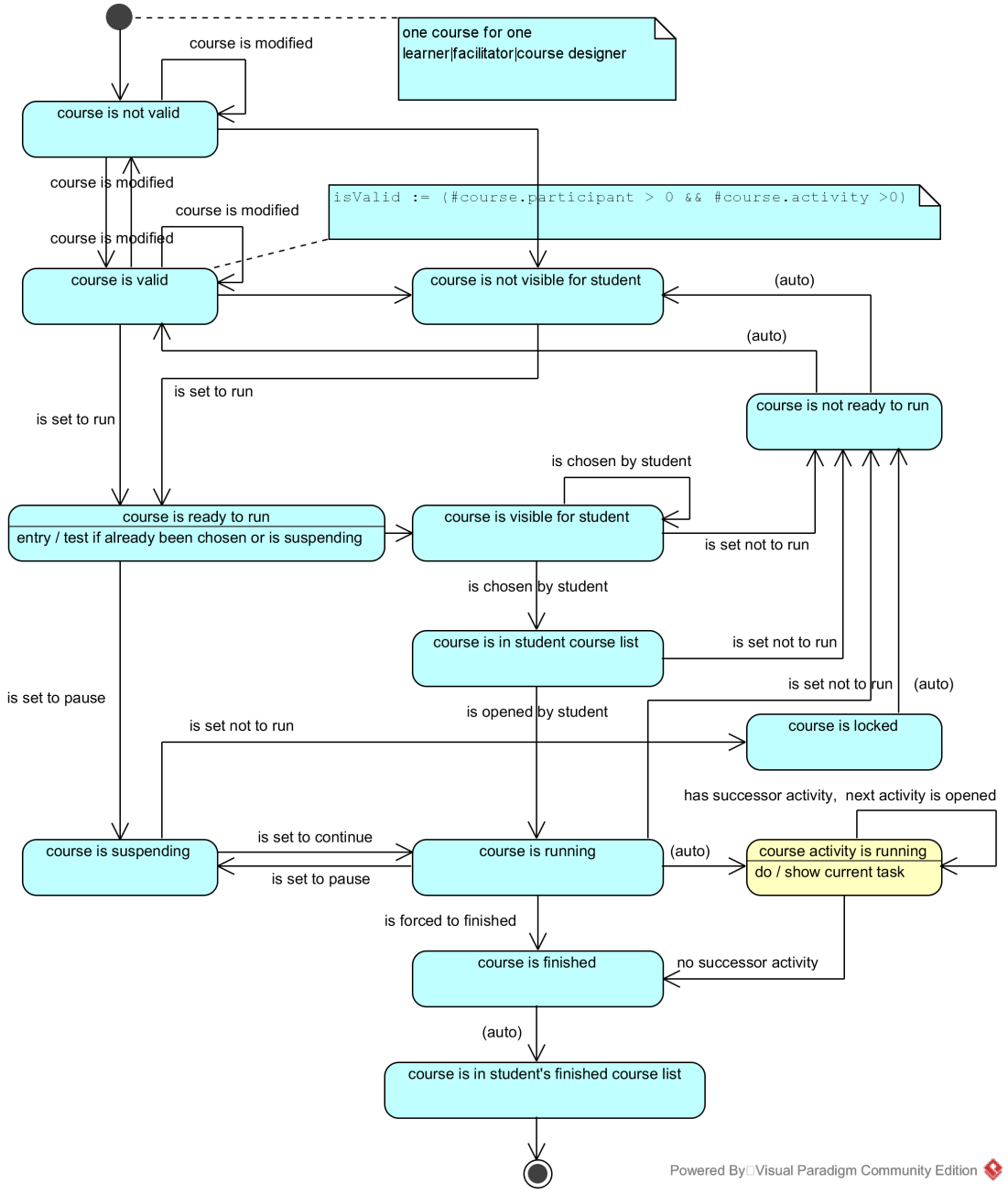


Figure 6.4: State-transition-diagram of the script instance runtime logic.

The start point was when a course was still not *valid*. A course that was not *valid* meant the arrangement (including arranging real world users to model, setting learning resources and other parameters to model, etc.) performed by an administrator of a course (PBL script instance / PBL model instance) was not finished.

The course would stay *invalid* when the arrangement (modifying) was underway. If the design work was finished, then the arrangement was finished. Of course, if the administrator did not satisfy the arrangement, the course could return back to the *invalid* state.

A *valid* course did not mean the course was visible for students unless it was set to *ready-to-run*. After a course was set to be *ready-to-run*, students could view and agree to take part in it. The course could then be opened by students and start the problem-based learning process under the model definition. A course that was opened meant that the course was in the state of *running*. However, even if a course had been started, it could still be suspended if the administrator set the course to *suspending*. Then the course was clocked, and students and facilitators could not use the course for PBL anymore. *Suspending* did not mean the course was closed; if the teacher set the course to continue, then the course could be restated from the suspending point rather than from the very beginning again. On the contrary, the administrator could also set the course to not run, which would cause the course to be closed, no matter if the former state was *ready-to-run*, or was chosen by students, or was in the *running* state, and the course would be closed immediately and could only be restarted from the very beginning.

If a course was in the *running* state, then the activities of the course would be interpreted one by one based on the script model. The interpretation will be presented in next section in detail. If all activities were performed and no successor activity could be found, then the course state would be changed to the *finished* state. Meanwhile, students could not see the course from their *ready-to-run* course list anymore, which meant a complete course state transition cycle was finished.

6.3.2 Runtime Logic in Engine

Section 6.3.1 presents the course state transition, namely the state transition of PBL script instance. However this was just a high-level description of the runtime logic of a course (script instance) as the core composition of a course—learning activities—were not mentioned. In this section, I used a flow chart to present the learning activities runtime logic. Figure 6.5 illustrates the runtime logic of the *PBL Script Instance Interpretation Service* (Course Runtime Engine) and the relationship with the *PBL Course Management Tool* and the *PBL Online Whiteboard*. This figure reflected the major system-user interaction of the *PBL Runtime*

depicted via Figure 6.2 and Figure 6.3.

As we can see from Figure 6.5, there were four columns. These columns separately indicated that the *Script Instance Management* interacted with an administrator, *Script Instance Interpretation Service* (Course Runtime Engine) was for the system, *Course Management for Student* listed the available courses and the corresponding problem-based activities, and the *PBL Online Whiteboard* carried out learning activities through a Web-based whiteboard like UI for students.

The first column, *Script Instance Management*, roughly showed the course state transition described in the last section. Therefore the start point in the context of this section was the red point. When a course was set to ready for running, the state of the first course activity would be set to *ready-to-run* at the same time. Since the course was *ready-to-run* for running, the involved participants would see the course (the third column), then they would click the start-button on the listing user interface to open the *PBL Online Whiteboard* to start the learning interaction (play a course in the four column). Before the whiteboard was finally opened, it would ask the engine if there was an activity still running (e.g.: sometimes when a user accidentally closed the whiteboard, the corresponding activity would stay in the running situation), he or she should be able to reopen the activity, meaning this check was important). Then:

- If there was an activity still running, the engine would tell the whiteboard the activity retrieval information. With this information, the whiteboard could ask the engine to give it the corresponding activity body, including involved actors, operation permission policies, learning information resources, learning tools, expected learning outcomes, etc.
- If there was no running activity, the engine would find the next *ready-to-run* activity. If the course was just started at this very moment, the next *ready-to-run* course activity would be the first learning activity. Otherwise, the next *ready-to-run* activity would be the second one, third one, and so forth). If there was a next ready-to-run activity, then the engine would tell the whiteboard the activity retrieval information; with this information, the whiteboard could ask the engine to give it the corresponding activity body as described above. Meanwhile, this course activity would be set to running and the next course activity would be set to *ready-to-run*.

No matter if the activity was *running* or *read-to-run*, the whiteboard would render all the activity body information for the learning participants in the same way.

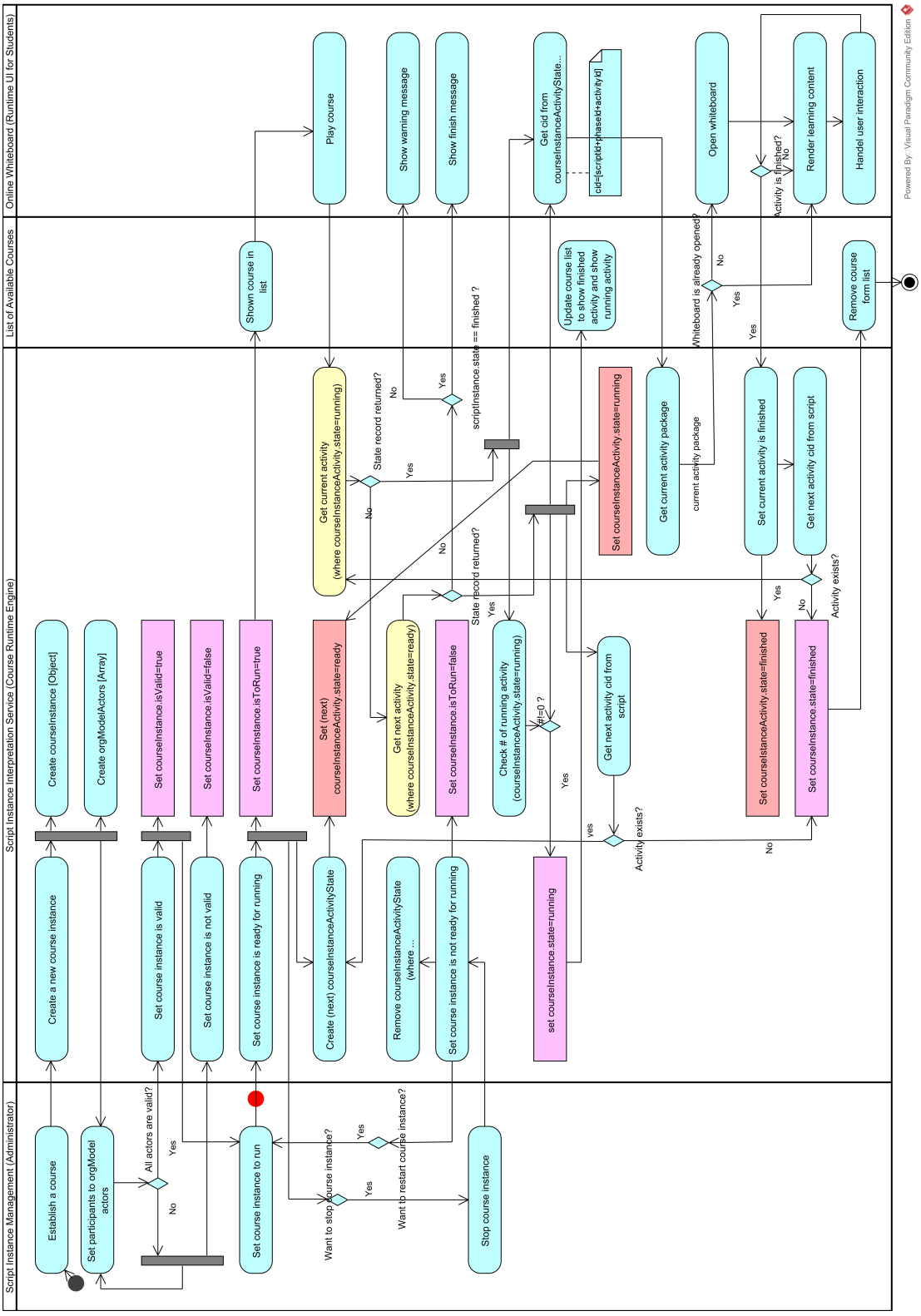


Figure 6.5: The runtime logic of the *PBL Script Instance Interpretation Service* (Course Runtime Engine) and the relationship with the *PBL Course Management Tool* and the *PBL Online Whiteboard*.

What the whiteboard needed to have was an activity interpretation function and a user activity handling function. Several ways could finish a PBL learning activity. Generally speaking, it could be automatically finished when a certain condition was satisfied, or be forced finished by facilitators or any other participants who had the activity termination permission.

When an activity was finished, then the whiteboard would send the message to the engine. The engine would then set the current activity state to *finished*, which also meant an activity runtime cycle was finished and a new cycle could be started by preparing the next *ready-to-run* activity. If there was no next *ready-to-run* activity, then it meant the whole course was finished, which bring us back to the course stat transition level.

Additionally, the *ready-to-run* activity of a script was also a complex work that involved the diagram interpretation of the learning process (workflow process). Although this work was also crucial for the system, considering the length of this thesis, I don't present this level of detail.

6.4 Manipulation of PBL Model Data

The previous section explained the way to meet the requirements of computer supported PBL from the modeling point of view. However, before actually implementing it, there were still several technical difficulties the needed to be overcome in order to implement the MDA. The difficulties included:

- How to represent and handle the networked graphical actor organization and the multi-level structured learning process in the design time, especially when the depth and the degree of the representation were uncertain.
- How to effectively operate the model elements where the operation included save, retrieve, interpret or transform all the elements, all the properties of every element, all the relationships between the elements, etc., especially when the elements were constantly changed with different levels of model abstraction.
- How to transform a multi-level graphical process to a complete process script or a textual document.
- How to interpret the script to a PBL module for the run-time environment.

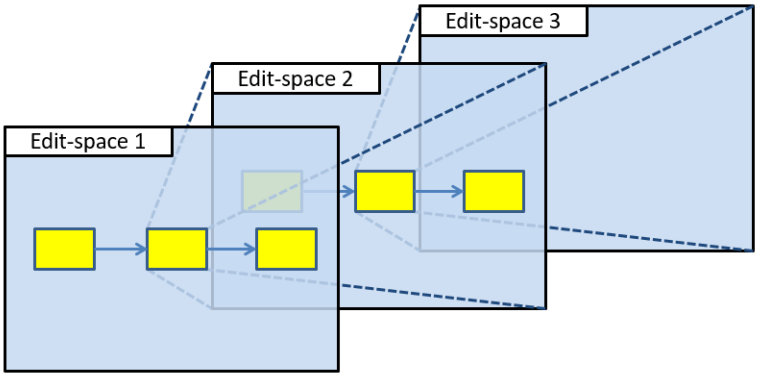
All these difficulties were caused by the fact that the structure of the process models and the element information inside the models in our application were semi-structured.

6.4.1 Handling Semi-Structured Process Models

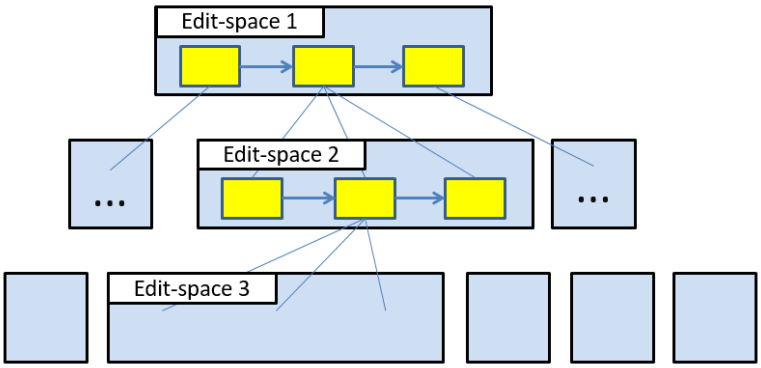
According to (Buneman, 1997), for the semi-structured data, either there was no separate schema to constrain the information, or the schema existed but only implied loose constraints. Under this description, the PBL process models were semi-structured, since there was no schema to constrain the depth and degree of the processes. Although the models were semi-structured and very flexible, they still had a homogeneous hierarchical characteristic: every next level in a model was the definition of an element in the previous level, and within the same level, the information was only about the properties of the elements and the relationship between the elements. Therefore, based on this characteristic, the authoring workspace could also be designed to support the representation of the PBL processes hierarchically, which meant each edit space (the most important component of the *Adaptive Flowchart Design Style UI System* introduced in Section 6.5.1) represented the definition of the upper layer element. The definition of each layer was a directed graph that consisted of elements and connections.

Figure 6.6a and 6.6b illustrates the hierarchical process representation described above from two aspects. For example, edit space 1 was for designing the phase sequence. After a phase was defined, it could be opened into a new edit space. Similar to the first level, in edit space 2, activities, resources, tools, actors, artifacts, and their relationships could be defined. If required, the third level of edit space could be opened. Therefore, a process script even with infinite depth or degree was possible to be defined and represented. Since the authoring operation requirements in each level were similar, the functionalities required for each edit space was also similar.

Correspondingly, the database could also be designed to save or retrieve the process elements and the relationship iteratively like the representation manner in order to handle the infinite depth and degree of the script data. Because there were only finite meta-element types (phase, activity, resource, tool, actor, and artifact) in the metamodel (the PBL scripting language) based on the design of our MDA, all elements could be saved or retrieved in their own meta-element types of collections.



(a) Perspective A.



(b) Perspective B.

Figure 6.6: The hierarchical representation concept of a PBL process in multi-level edit spaces.

Relationships among elements were seen as connection element that could be saved or retrieved in a connection collection.

Figure 6.7 illustrates this kind of data saving and retrieving approach. Specifically, each element had a definition id that pointed at another edit space; each edit space had a segment id to let the elements and connections know which edit space they belong to. Elements and connections of different edit spaces from different scripts were stored together in their own meta-element types of collections. The retrieving of a definition was to find all the elements and connections where their segment id equaled to the edit space's segment id. Then, the data could be directly sent back without any process logical calculation. Based on their graphic coordinate information, all elements and connections would be correctly put back in the corresponding edit space. When the designer wanted to view the complete script, all the elements and connections could be easily retrieved because all those who belonged to one script had the same script id. The only additional step was

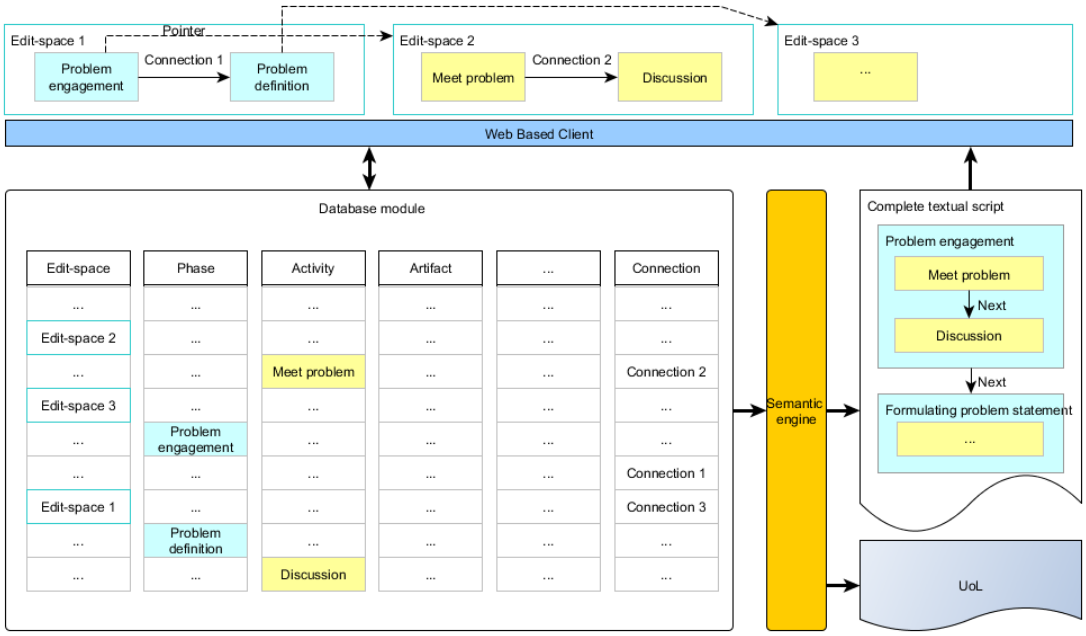


Figure 6.7: The storing and retrieving concept of PBL scripts.

that a semantic engine would assemble them into a correct order; also because of this engine, the authoring tool was able to produce the script as a UoL for IMS-LD compatible players as well.

Because the elements and the connections were stored separately, it showed several benefits:

- It was highly efficient to add, remove, update, and find one element wherever it was in one or multiple PBL scripts. When a designer edited an element, changes could be updated to the target element without searching all the elements of all models. This benefit was effective for reducing the computation cost in semi-structured data searching.
- It was very easy to handle the change of the process structure. Because the elements were not nested to each other, the change of structure only affected the change of the definition id and the connection element. This benefit made it possible to effectively update, reuse, and share any level of sub-process among the PBL process scripts.

6.4.2 Handling Semi-Structured Data in the Process Models

The previous section elaborated the approach to handle learning processes. However, not only was the structure of the PBL process semi-structured, but also the element information itself was semi-structured.

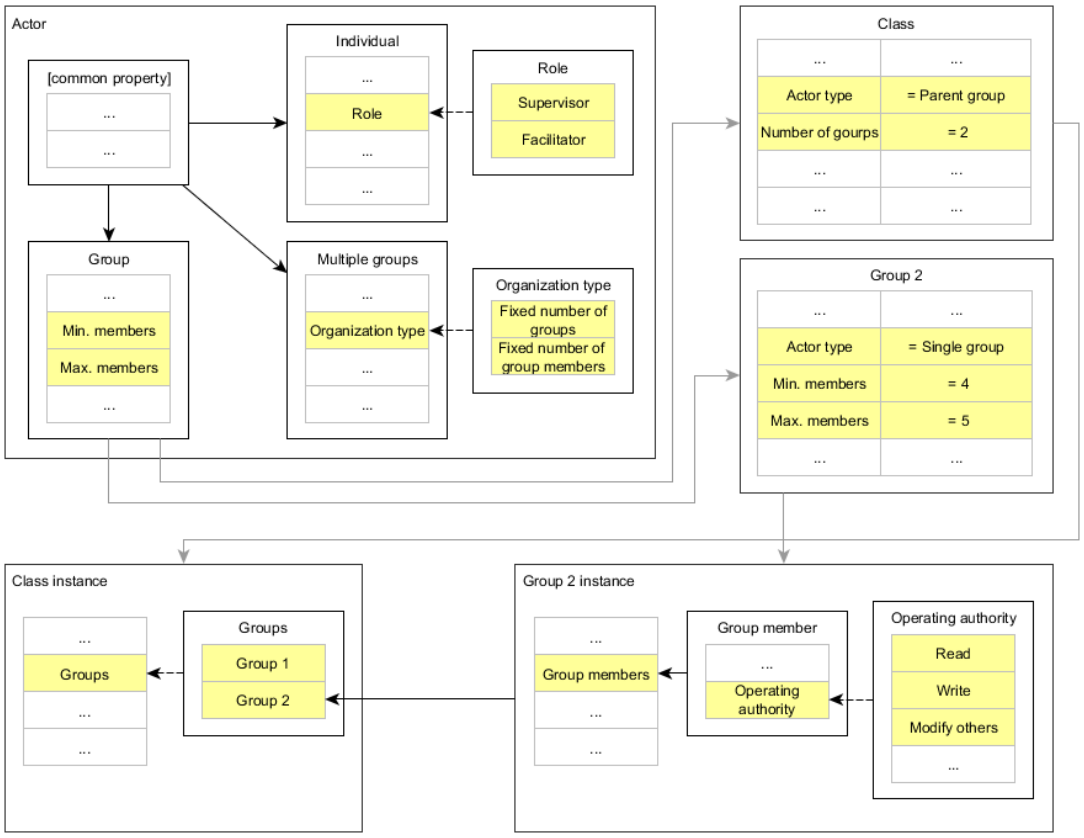


Figure 6.8: Constantly changed properties of model elements in different levels of models.

As illustrated, our system applied MDA, and the elements of models were constantly changed in order to adapt different levels of process abstraction during the model transformation. Therefore, from the process element viewpoint, as an example, the change of information inside the elements could be shown in Figure 6.8. In this diagram, we used the actor element to show how the properties tended to be constantly changing in different modeling stages. When the actor element belonged to the metamodel, the application needed to store all the property definitions (and type definitions and relationship definitions) and the corresponding option values. For example, according to the scripting language, the meta-element

multiple groups had a single choice property named organization type, and it had the options of a fixed number of groups and a fixed number of group members.

However for the meta-element group, it did not have this property. For example, it had other properties such as *minimum members* and *maximum members*. Furthermore, although a *Class* element or a *Group 2* element was created based on the same meta-element group, the property fields were different. According to the storing strategy illustrated before, these two elements needed to be saved in the same data collection: the actor collection. Also because of this difference, when the *Class* and *Group 2* were instantiated, the instance results were consequently quite different with each other. As shown in Figure 6.8, *Class* instance needed to save the grouping information, while the *Group 2* instance only concerned the group members' information and their operation policy. In summary, the number, depth, and types of the properties needed to be stored in a very flexible manner.

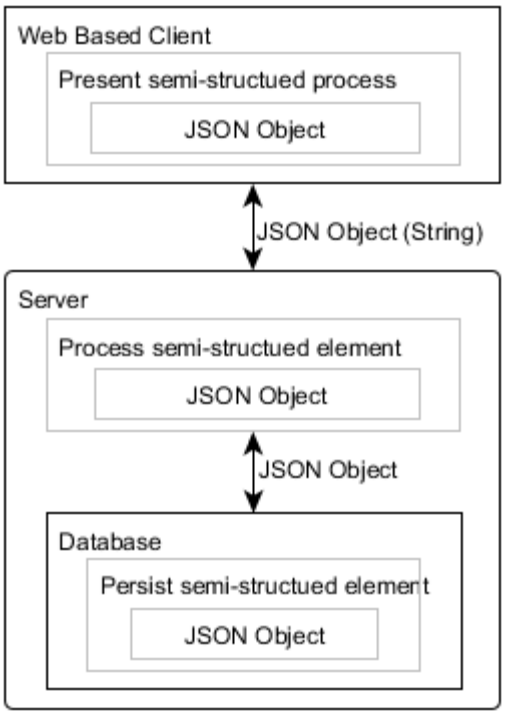


Figure 6.9: System architecture from the perspective of the JSON data type.

As a Web-based application and also considering to avoid browser compatibility issues and realize the semi-structured flexibility, we had chosen Node.js as our web server and JavaScript and JSON to build both client-side and server-side modules. Also, we used a NoSQL database to handle the semi-structured data storage and retrieval. The reason why this combination of technologies was effective to handle

the semi-structured element data was that JSON was an ideal data-interchange language for representing semistructured data, and it was a subset of the object literal notation of JavaScript. The element and connection objects in the application could simply be JSON objects.

Consequently, the client side and server side could communicate with each other directly through a JSON object (after serialization), which meant that on the client side, elements and connections could be directly rendered to edit space or be sent back to the server side without any format transformation. Also, without any transformation, the server side could directly process the elements or connections from the client side. Alternatively, data from the client side could be stored in the database directly, even if there were different numbers, depths, and types of the properties in the same type of object. For example, the client- and server-side JavaScript interpreter and the MongoDB, a NoSQL technique, could handle and store the element *Class* and *Group 2* as a JSON object natively. Figure 6.9 shows the system architecture from the perspective of handling the semistructured element information. Driven by the PBL scripting language, together with using pure JavaScript for the client UI, Node.js as the Web server, MongoDB for the data persistence, JSON as the unified semi-structured data format in the whole system, and with the approach for handling the semi-structured process models, all the technical requirements for implementing an application that were able to support the authoring, delivering, and execution of PBL processes were met.

6.5 Component Structure of IDEE4P

The previous sections presented the IDEE4P system from the high level conceptual design to the relatively low level specific logic and data manipulation design. As a summary of all of this, this section will present a component structure diagram to finish this chapter.

Unlike Figure 6.2 showing the functional architecture of the IDEE4P, Figure 6.10 shows a component structure of the IDEE4P from the perspective of computational logic and service. This diagram emphasized two components by giving a relative bigger area and more components inside. These two components were the *Web-based UI of PBL Authoring and PBL Language Editing* and the *System Back-end*. Secondary components included a *Web-based UI of Course Management*, a *PBL Online Whiteboard* (the native PBL Player), a *Script Textual Report*

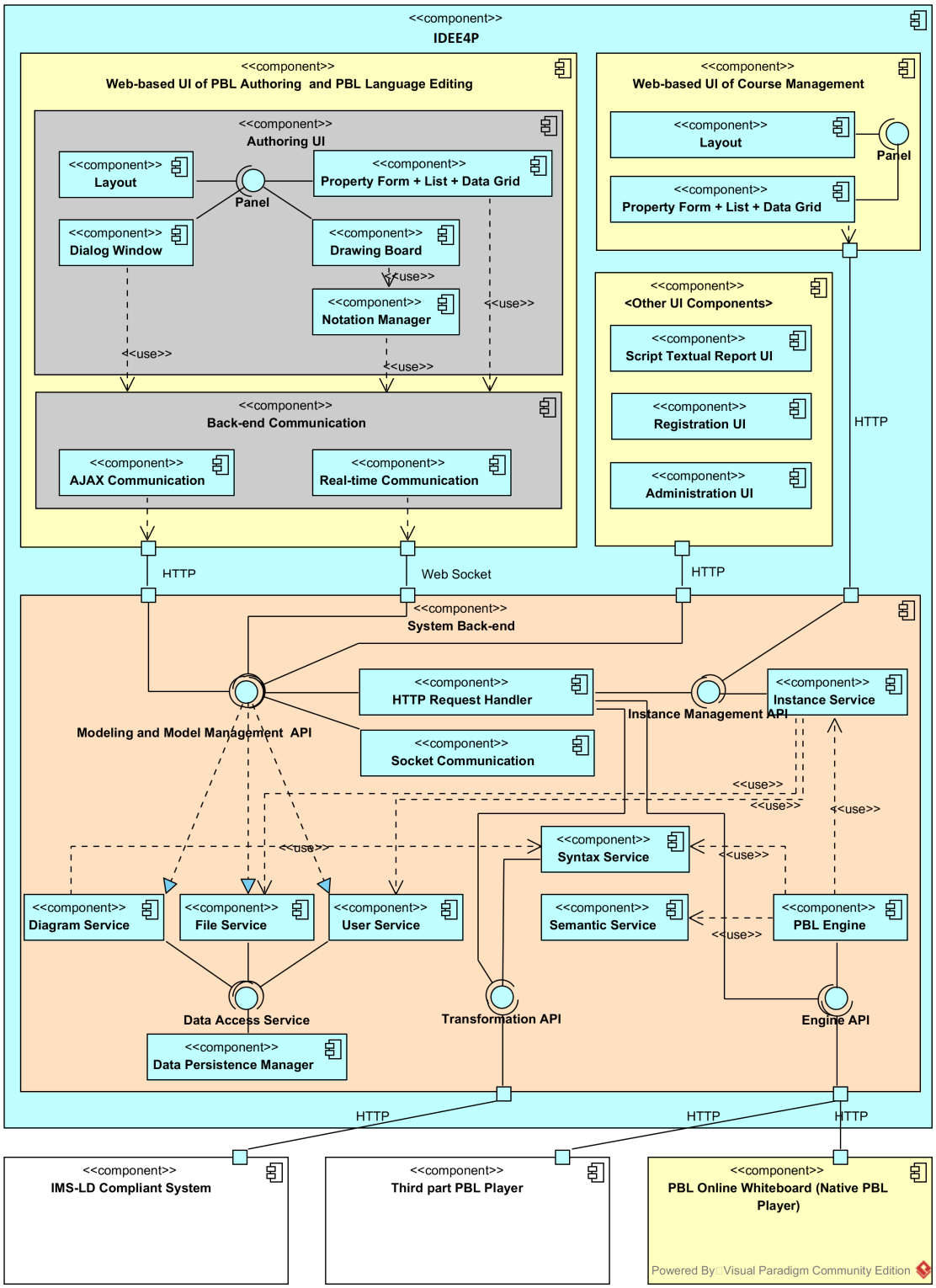


Figure 6.10: A component structure of IDEE4P from the perspective of computational logic and service.

UI, a *Registration UI*, and an *Administration UI*. Other even smaller components were ignored in this simplified component structure diagram. Additionally, the *PBL Online Whiteboard* was seen as a secondary component did not mean it was not complex in technical design and realization. It was because the *PBL Online Whiteboard* was designed at the level similar to IMS-LD players and was seen as an external service target of the core components of the IDEE4P, although it was developed natively for the IDEE4P. In fact the internal component structure of the *PBL Online Whiteboard* was also complex. By realizing it, a thoughtful design was needed and many technical challenges were present. However, as this was not the focus of this thesis, I intend to avoid presenting too many details about it.

6.5.1 An Adaptive UI System for PBL Authoring and Language Editing

By presenting the basic architecture of the IDEE4P in Section 6.1.2, it was mentioned that the *PBL Language Editor* and the *PBL Authoring Tool* shared the same *Adaptive Flowchart Design Style UI System*. In the adaptive UI system, I used an *Ext JS*[1] technology to provide a layout container. By using the panel interface of this container, the UI system could, to a modeling context, adaptively bind a *Drawing Board* component, a *Property Form* component, a *Data List* component, a *Data Grid* component, and a *Dialog Window* component. Particularly, the *Drawing Board* component further depended on a *Notation Manager* component.

The basic working principle of this *Adaptive Flowchart Design Style UI System* was, for example, when we were going to edit a phase-activity process, the UI system would know that the workspace should be multiple layers rather be than a single layer for PBL language or actor organization structure editing. Therefore different panels, such as the script file management panel, property editing panel, toolbar, operation menus, etc., and a phase-activity process specific drawing board would be rendered. When users operated a language element, such as phase nodes, activity nodes, or the connection edge between elements, the *Notation Manager* would determine the operable property list or check if the current operation was valid. This determination and check management was based on the definition of

[1]Sencha Ext JS includes the industry's most comprehensive collection of pre-integrated and tested high-performance UI components. These components include HTML5 calendar, grids, pivot grid, D3 adapter, trees, lists, forms, menus, toolbars, panels, windows, and much more. For a more comprehensive list, visit https://www.sencha.com/products/extjs

language model (or script metamodel) in the context of PBL authoring or language metamodel (or script metametamodel) in the context of PBL language editing.

In the *Adaptive Flowchart Design Style UI System*, there was also a *Back-end Communication* component responsible for sending data to or getting data from the remote server. Components such as the *Drawing Board*, *Property Form* component, *Dialog Window*, etc. havd modification listeners that could catch all the user's editing or authoring operations and tell the *Back-end Communication* component about the change. The communication component used either an *AJAX Communication* asynchronously or a *Real-time Communication* component synchronously to send the changes. Moreover, the *Real-time Communication* component was also responsible for receiving changes from back-end server side to update the editing or the authoring UI. This component was important for collaborative editing or the authoring.

6.5.2 System Back End

The system back-end was relatively complex. It consisted of a *HTTP Request Handler* component, a *Socket Communication* component, a *Diagram Service* component, a *File Service* component, *User Service* component, a *Data Persistence Manager* component, a *Syntax Service* component, a *Semantic Service* component, a *Instance Service* component, and a *PBL Engine* component. Although there were four different tools, they shared a single same back-end. The responsibility of each component in the back-end was explained as the following:

HTTP Request Handler handles all HTTP requests from the Web-based UI in the PBL authoring, PBL language editing, course management, textual script report, registration, or system administration. It used the interface provided by the *Diagram Service*, *File Service*, *User Service*, *Instance Service*, *Syntax Service*, *PBL Engine* to forward the system operation request to the corresponding service. For example, if a user added an activity onto a phase-activity drawing board, the request handler would forward the data of the activity to the *Diagram Service*. This component transformed the data to be ready for computation and finally used the *Data Persistence Manager* to store the data.

Socket Communication handled similar work as the *HTTP Request Handler*. The major difference was that it sent back data back to the UI in a real-time

manner.

Diagram Service handled the workflow chart of models, no matter if the models were metametamodels, metamodels or models. For example, if a user updated a property of a model node, this service would find out the property of the node and update necessary information to the model. The component depended on the *Syntax Service* to understand the structure of a workflow diagram.

File Service saw every model as a single file. However, a file also had its information of category, display structure, etc. This component handled the perspective work for models. Moreover, the component was also responsible for the management of uploading other resources, such as images, etc.

User Service was responsible for all types of system user related work, such as user registration, role management, permission management, etc.

Data Persistence Manager realized the work of model storage and retrieval. The major work of this component was described in Section 6.4.

Instance Service realized the work of script instance management. The primary work of this component was described in Section 6.3. This component depended on the *File Service* to handle scripts and the *User Service* for the user arrangement.

Syntax Service was responsible for the telling system how to correctly process models. For example, to indicate how many levels a script had, which element could have which kind of incoming edges, which elements needed to be there for what model, etc. This component was embedded with a predefined PBL metametamodel and a changeable PBL metamodel, which was defined by experts.

Semantic Service was responsible for model interpretation. It depended on the *Syntax Service* component. The interpretation was based on the external input as state transition triggers. The state transition design for course and course activity was presented in Section 6.3.

PBL Engine handled all the HTTP requests from the *PBL Online Whiteboard* or third party PBL players. It was also responsible for packaging activity information. This component was a manager of *Semantic Service* and *Instance Service* in order to expose the script runtime operation API as simple

as possible.

Other components, such as *Web-based UI of Course Management*, *Script Textual Report UI*, etc., and the way how the IDEE4P interacted with external systems or tools, such as *IMS-LD Compliant System*, was also illustrated via Figure 6.10.

Chapter 7

Implementation

This chapter is particularly based on the publication "A Model-Driven PBL Application to Support The Authoring, Delivery, and Execution of PBL Processes" by Wang, D., Samaka, M., Miao, Y., Ali, Z., and Hoppe, H. U. (Wang et al., 2016)

In this chapter, four tools will be shown as the implementation result based on the my technical design illustrated in the last chapter. Those four tools include a *PBL Authoring Tool*, a *PBL Course Management Tool*, a *PBL Runtime Player*, and a *PBL Language Editor*. From the perspective of my research, those four tools can be divided into two groups: The PBL Authoring Tool belongs to the first group; while the PBL Course Management Tool, the PBL Language Editor and the PBL Language Editor belongs to the other group. It is necessary to emphasize that the implementation of the PBL Authoring Tool in the first group was my major work, because the Learning Design authoring was the most important component in the IDEE4P to facilitate the problem-based Learning Design. However, it was still necessary to implement the second group of tools. There were two major reasons: (1) Because my model-driven approach involved multiple layers of tools, it was meaningful to provide an complete example to show how other tools were integrated with the authoring tool. (2) It was worthwhile to realize a relative complete system, so that we could have a better test environment to the evaluation of the authoring tool.

7.1 PBL Authoring Tool

This chapter will illustrate the major functionalities being implemented in detail through carrying out the example scenario Deformed Frogs. Figure 7.1 shows the typical user interfaces of the *PBL Authoring Tool*. These screenshots were captured when we presented the deformed frogs learning scenario. As illustrated by the functional architecture, the authoring tool consisted of two authoring editors: an actor/group structure editor and a phase-activity structure editor (or learning process editor). In Figure 7.1, from top to bottom, the three screenshots correspondingly indicated the organization modeling in the actor organization editor and the learning phase process authoring and the learning activity process authoring in the phase-activity process editor.

7.1.1 A UI for Actor/Group Structuring

All the three editors provided a similar user interface with operation-alike functional components. The similarity was intended since similar user interfaces could reduce the difficulty of using the three different tools. This ensured a higher usability from a certain aspect. If we looked into each editor, the middle parts were both a graphical diagram-alike workspace that enabled designers to structure the organizational model or the phase-activity process model more easily and freely. Each of the graphical workspaces had a dynamic list of building blocks at the top-left corner. The dynamic building blocks were context appropriate for building the actor organization or the phase-activity process. On the left of the workspace, a process-script file management panel was provided for handling the created process scripts for each PBL designer. On the right, there was a context-aware property editing panel that is used to show and set the properties of the selected actor or phase-activity element in the workspace.

When modeling the actor organization through the organization editor, the designer could simply drag the "actor" icon from the building block list and drop it onto the graphical workspace to create an actor. The application would tell the designer that the type of actor could be *Individual*, *Group*, or *Multiple Groups*. Theoretically, these three types of actors were sufficient to make the designer built a complex enough organization model. As shown in the first screenshot, we build a very simple class organization model. In this model, an individual actor was created and named as "Facilitator;" a group actor was created and named as "Class."

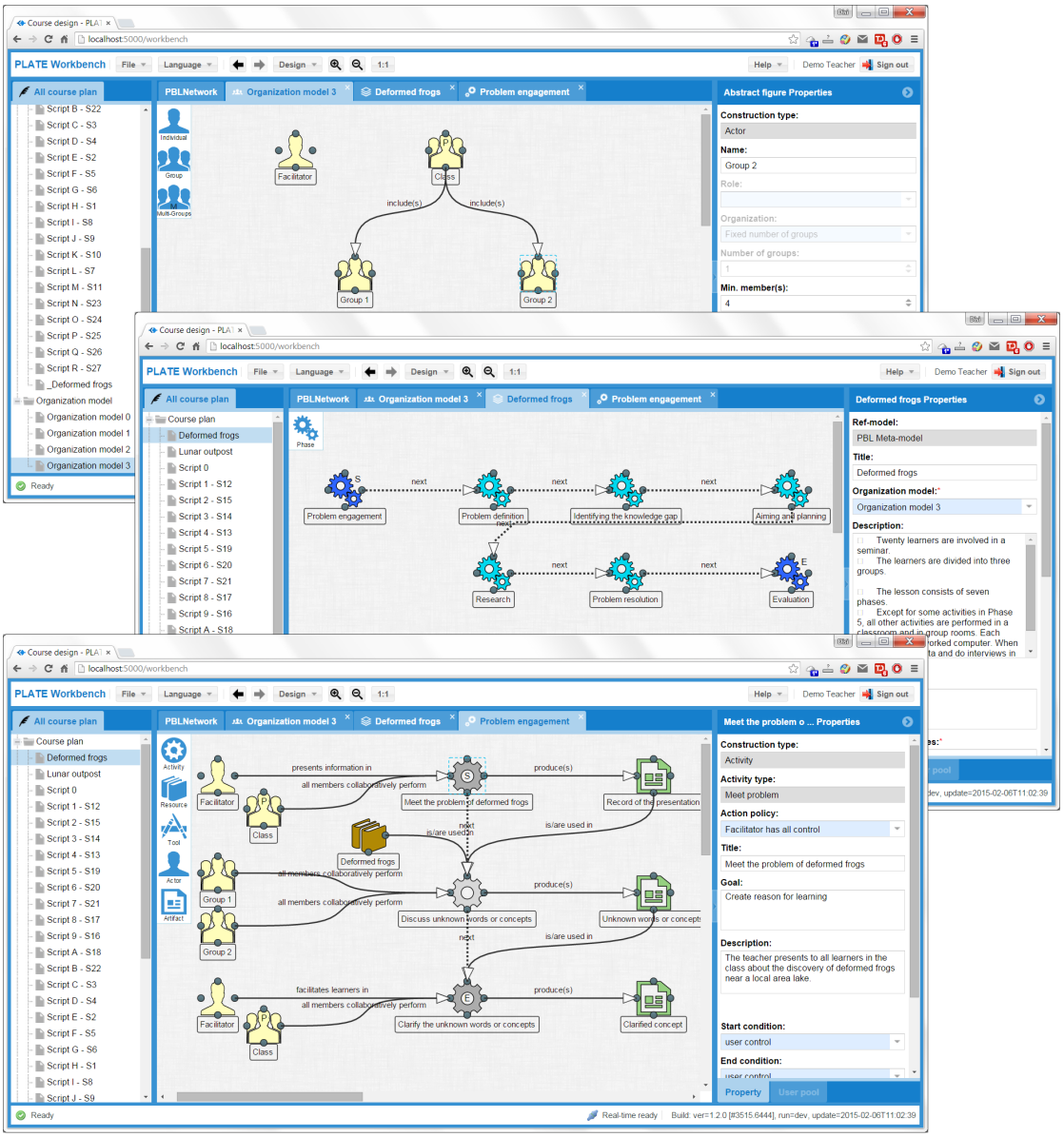


Figure 7.1: Screen-shots of the *UI for Actor/Group Structuring* and the *UI for Phase-Activity Structuring* in the *PBL Authoring Tool*.

Two sub-groups, “Group 1” and “Group 2” were added underneath this class. In the property panel, the “Group 2” was specified to have a maximum of four participants. The designer then created directed connections between the class and the groups. The connection between actors was important in defining the element relationship for the organization. With the connections, the “Class” became a parent group of both “Group 1” and “Group 2”.

Along with the organization structure authoring, there was another very important editing function: the permission policy settings of the actor role for the operation

in the runtime learning environment. Figure 7.2 shows the setting dialog.

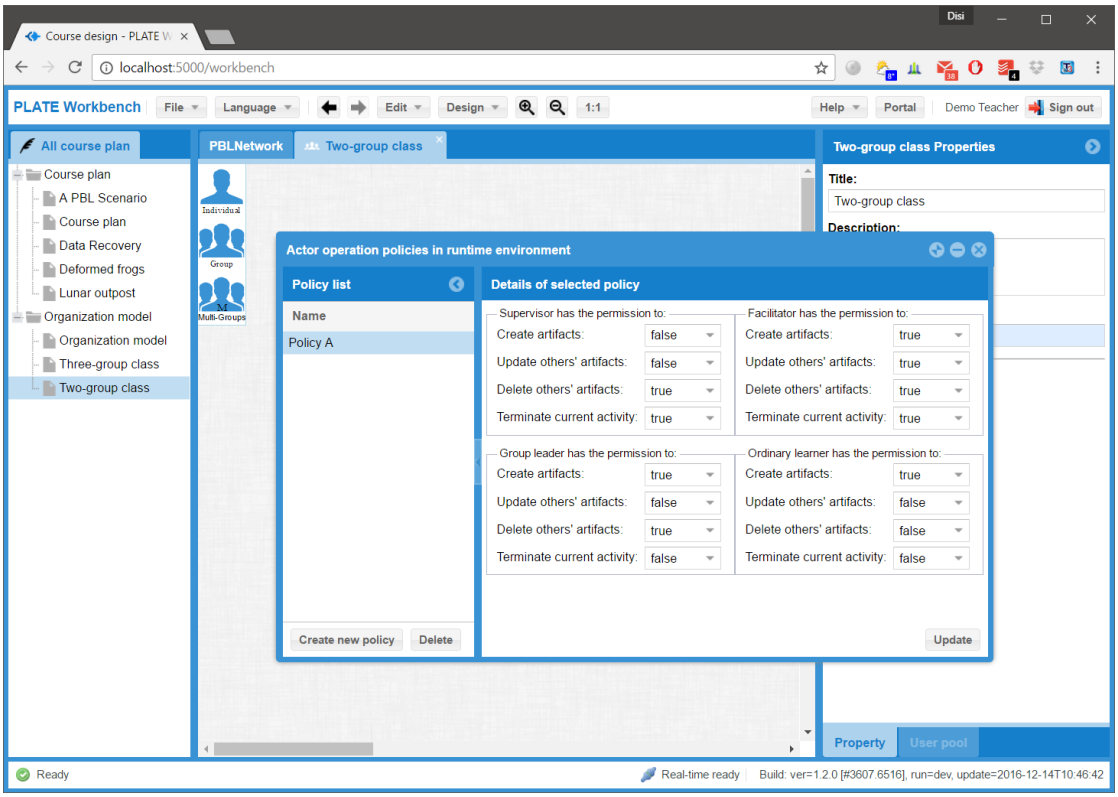


Figure 7.2: Permission policy settings of the actor role for the operation in the runtime learning environment.

A learning organization could have several different permission policies. Each policy could be set upon different focuses. A policy involved four different action roles; they were: *Supervisor*, *Facilitator*, *Learning Group Leader* and *Ordinary Learner*. Each action role had four different operation permission settings affecting runtime activities. The settings included:

1. Ability to create learning artifacts or not?
2. Ability to modify other people created artifacts or not?
3. Ability to delete other people created artifacts or not?
4. Ability to terminate the current running learning activity or not?

As an example for completing the actor organization editing, in Figure 7.2, the *Facilitator* was set to be able to create learning artifacts, modify, and delete other people's created artifacts, and terminate the current running learning activity. On the contrary, the *Ordinary Learner* was set to be able to only create learning artifacts for himself or herself. A more concrete explanation about how the permission

setting affects the learning activity running will be introduced in Section 7.2 *PBL Course Management Tool*.

7.1.2 A UI for Phase-Activity Structuring

Similar steps were performed when designing phase-activity processes through the learning process editor. The second screenshot in Figure 7.1 shows a PBL phase process model that was made up of several phase elements with several directed connections indicating the process sequence. The third screenshot shows the internal process structure of an upper-level phase element. Here, the three ordered activity elements were associated with other types of elements such as actors and artifacts. In comparison with the actor organization editor, there were more element types such as *Phase*, *Activity*, *Resource*, *Tool*, *Actor*, and *Artifact*, while only the *Actor* element type was available in the actor organization editor.

Roughly speaking, the process editor scaffolded the phase-activity process design from two perspectives. First, at the moment a designer put a phase element onto the workspace, a pop-up window listed all available phase types. These types came from different PBL frameworks, models, and practices. For example, from the type list, the designer could find the phase type, such as *Clarifying terms and concepts* or *Formulation of the problem statement*, to create a "Seven jumps" model-based phase; he or she could also choose the phase type, such as *Dependency and inclusion* or *Counter-dependency and fight*, to create a "five-stage" model-based phase. Then, when creating activity elements under a created phase, operations invalid or inappropriate to the current phase context were blocked.

As shown in Figure 7.1, the second screenshot represented the seven learning phases in order to describe the whole learning process of the deformed frogs at a higher abstraction level. Then, each phase could be opened like a folder to go to the lower abstraction level, which was shown in the third screenshot. In Figure 7.1, the third screen-shot depicted the detailed activity process (represented by the gray gears) of the first phase "Problem engagement" (created by choosing the phase type *Clarifying terms and concepts*). In this level of authoring, a designer was guided to create activities only with the activity types such as *Meet problem*, *Clarify concept*, *Discuss* etc. under the context of the phase type *Clarifying terms and concepts*. The same concept was applied for creating artifacts; options like *Record* and *Clarified concept* were found in the artifact-type list.

Actors in the learning process were the actors created in the actor organization editor. They could be assigned to activities as shown in Figure 7.1. Engagement modes could be set between actors and activities, where the engagement mode was the nature of the roles that the participants were expected or required to play while they were performing a learning task on the activity. For instance, when the designer assigned the class to the activity "Meet the problem of deformed frogs," as shown in the third screen-shot, he or she was guided to select one engagement mode such as *Each member individually performs*, *All members collaboratively perform*, and *Each group separately performs*. As we could see in the screen-shot, the *All members collaboratively* perform was chosen.

7.1.3 Textual Output of the Graphical PBL Representation

The diagram style graphical representation of a PBL process was handled as a single PBL script by the tool. The script was stored in a PBL script repository for the purpose of retrieving, sharing, and reusing, as well as be exported as a textual script file. The textual script exporting provided an alternative way that enabled the designer to review or share his or her design entirely as well as to let learning participants have an overview of the whole learning process. Figure 7.3 shows the textual script of the example deformed frogs. If the designer had not specified the *Goal* or *Description*, the tool would automatically generate a predefined text corresponding to the element type.

In this textual script, the top level contained the phases; then, each phase was detailed with its own activities. In Phase 1 "Problem engagement," we could see each activity has an automatically generated summary according to the actors, resources, tools, and artifacts involved with it. For example, the generated summary description of Activity 1.2 "Discuss unknown words or concepts" was:

> In this step, with the learning resource *Deformed frogs*, and with the learning artifact *Record of the presentation* from the previous activity, the activity *Discuss unknown words or concepts* will be performed by *Group 1 Group 2* as following: all members of the *Group 1* will collaboratively jointly synchronously perform the activity; all members of the *Group 2* will collaboratively jointly synchronously perform the activity. 2 artifacts *Unknown words or concepts* will be produced by

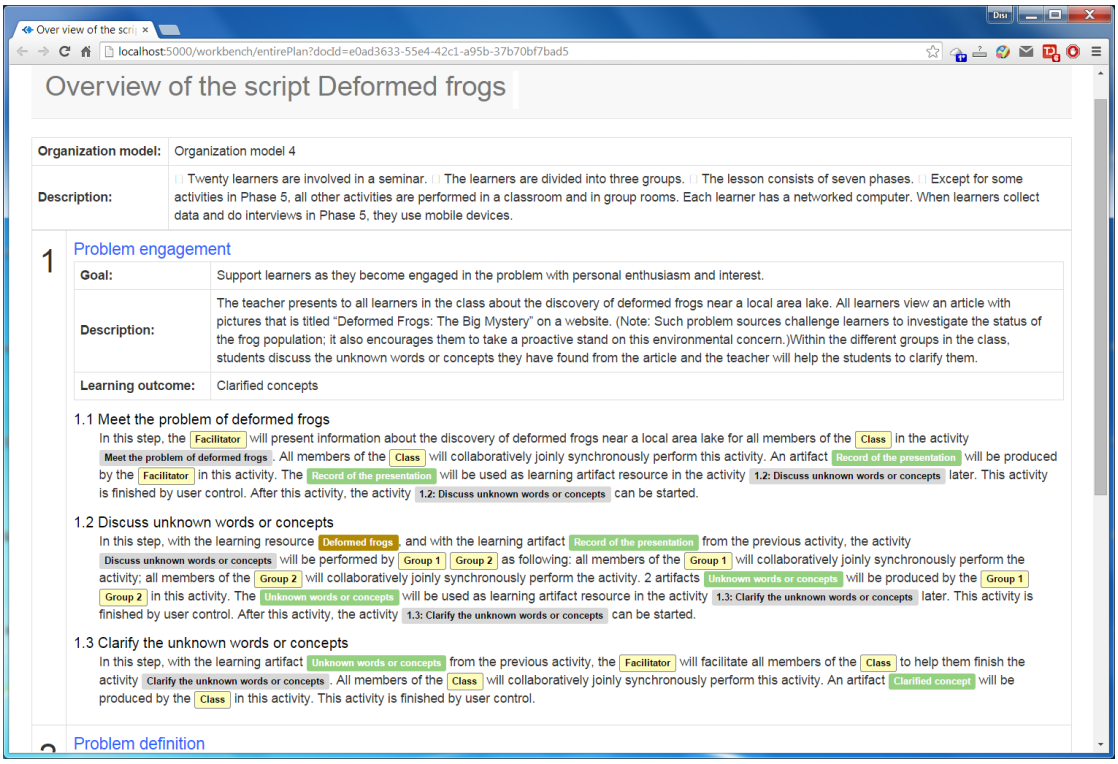


Figure 7.3: The textual output of the graphical represented PBL script (PBL process model).

> the *Group 1 Group 2* in this activity. The *Unknown words or concepts* will be used as learning artifact resource in the activity 1.3: *Clarify the unknown words or concepts* later. This activity is finished by *user control*. After this activity, the activity 1.3: *Clarify the unknown words or concepts* can be started.

This functionality showed that the tool had the capability of comprehending the meaning of the graphical process representation, or that the engine understand the semantic of the script and could help designers to transform their in-mind design idea into a computer- and also a human-understandable script. The tool was also designed to be able to translate the script into other scripts that could be run in other run-time learning applications, such as the IMS-LD compatible players as talked about in previous sections.

7.2 PBL Course Management Tool

Through providing an intuitive workflow diagram style graphical UI for the formal learning process representation, the *PBL Authoring Tool* makes PBL designers to express their PBL processes relatively easier. The graphical representations are stored as PBL scripts in the application. However, a script was actually just an abstract process model that did not come with concrete realizations. In other words, there were no particular learners, facilitators and learning resources linked to the activities of a model script, and the timely structure of the activities was still unavailable. This meant that the process was still not in a runnable state at this stage—it needed to be instantiated. For this purpose, a *PBL Course Management Tool* was specifically developed to support the management of the PBL script instance. In this paper, we refer to an instantiated PBL script (PBL process model) as a PBL course.

PBL Course Management Tool consisted of two different UI components: a *Course Administrator Management UI* and a *Course Student Management UI*. The layout of the *Course Administrator Management UI* basically consisted of three areas as shown in Figure 7.4. The left-side-panel listed all the previously generated process scripts. The middle part reflects the instantiation details of the selected learning process script. The middle part was split into two sections: the upper one section was used for the general settings of the learning process, while the lower section was intended for user management. The general settings were those properties related to the PBL module as a whole, while the user management part helped in actor management by assigning real registered users to the actor organization model of the learning process.

A learning process could have different general settings and different participants. On the right-side panel, the real registered users (system users, include teachers and students) were listed. Users could be added to or removed from the organization model. Their user information would be then added to or removed from the specific actor in the lower panel of the middle part. Once a user was added to the list of the participants in a particular actor, his or her context-appropriate role could be decided. For instance, a student-participant could have the role of an *Ordinary learner* or a *Group leader*, while a teacher-participant could be assigned a role such as a *Facilitator* or *Supervisor*.

As shown in Figure 7.4, the administrator had created two instances of the PBL

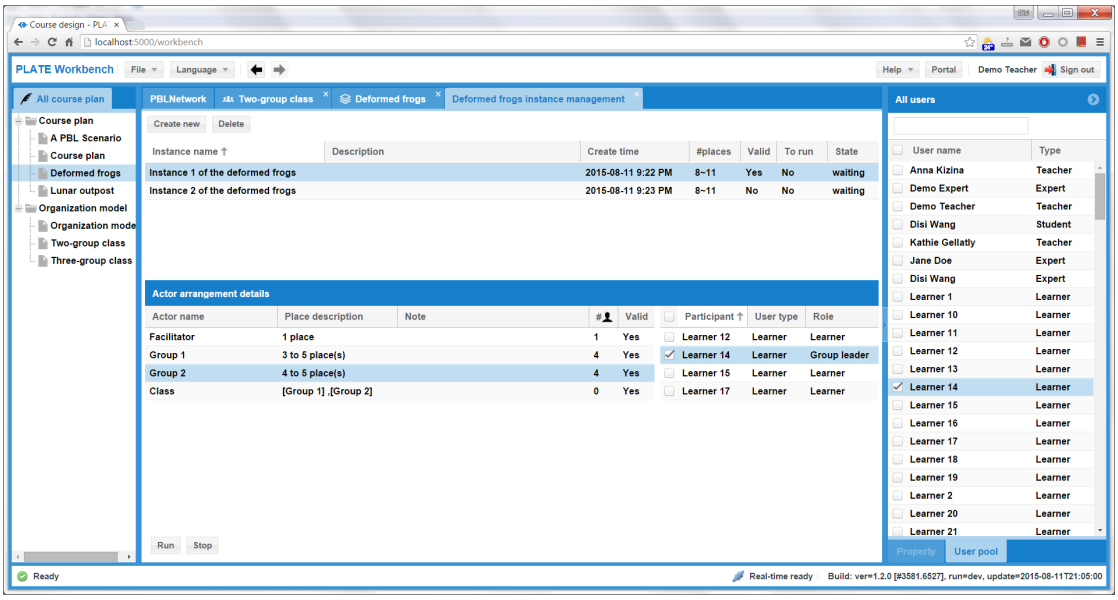


Figure 7.4: The UI of the *PBL Instance Management Tool*.

script "Deformed frogs." As an example, the course instance of the "Deformed frogs" was named as "Instance 1 of the deformed frogs" and "Instance 2 of the deformed frogs." In this screenshot, when "Instance 2 of the deformed frogs" was selected, the corresponding actors *Facilitator*, *Class*, *Group 1*, and *Group 2* were then listed in the actor panel below. In the actor listing panel, *Group 2* was selected and all the currently enrolled participants were shown right next to it. In the organization model, *Group 2* was set to have four to five participants. Once the number of the participants reached to four and there was not more than five, *Group 2* would be in a "valid" state. Furthermore, if and only if all the actors were in the valid state, the PBL course was in a "ready-to-run" state. This tool automatically delivered the instance to a PBL course repository after the module was *ready-to-run*. The state of the module would be changed to "running".

After the courses had their administrative settings set, involved participants, including students, and facilitators would be able to see the available courses (issued courses) from the *Course Student Management UI*, whose layout is shown in Figure 7.4.

The *Course Student Management UI* was responsible for publishing and updating the status information of PBL script instances (PBL courses) for different participants according to the arrangements. The *Course Student Management UI* provided an entry to let PBL facilitators and students be able to enroll to and only in these learning activities that they were arranged to engage in. Facilitators,

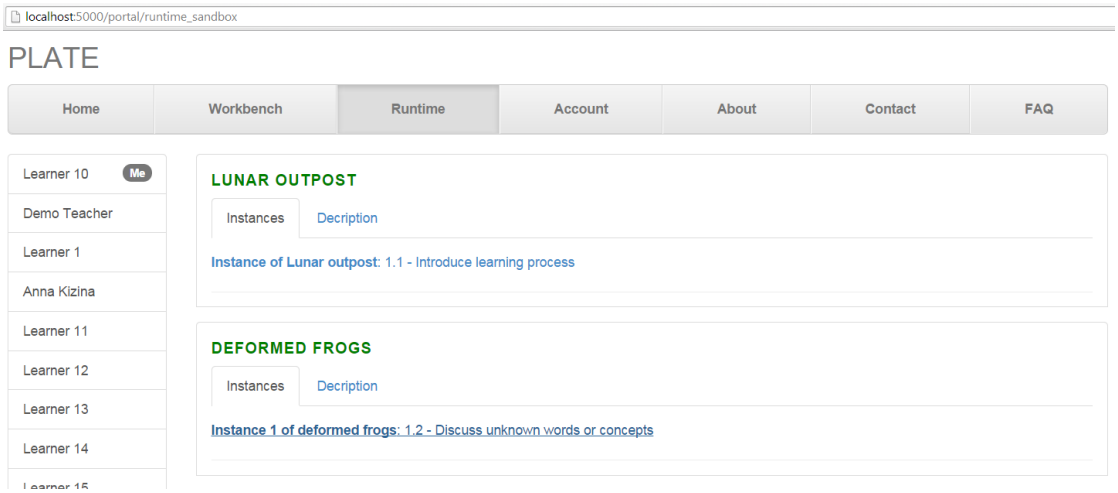


Figure 7.5: A screen-shot of the *Course Student Management UI*.

however, havd higher privileges, such as monitoring the activities and the progress made by students. The higher operation promotion made it easier for them to actively support learners in their learning activity. Section 7.1.1 has introduced this permission setting. All those settings would be carried out in the *PBL Online Whiteboard* - a native PBL runtime player of the IDEE4P. The next section will introduce the realization of this tool.

7.3 PBL Runtime Player

It has been shown that the problem-based learning process could be designed using the *PBL Authoring Tool* and instantiated through the *PBL Course Management Tool*. Currently, a mental problem-based learning process in a teacher's mind had become a computational runnable instance. Now, as we further introduce the *PBL Online Whiteboard* shown by Figure 7.6, the teacher's mind became a real interoperable file with his or her students.

The term "whiteboard" was often used in the PBL community as a three-column or four-column real solid whiteboard in the front of the classroom. In this paper, we extended this concept as a shared run-time learning space that could be used by individuals in blended or distance PBL courses (sessions or modules). The *PBL Online Whiteboard* was implemented in response to the fact that: (1) the existing learning script runtime tools were rare and not popular; (2) they were not designed with PBL in mind, and thus lacked support of the elements that were

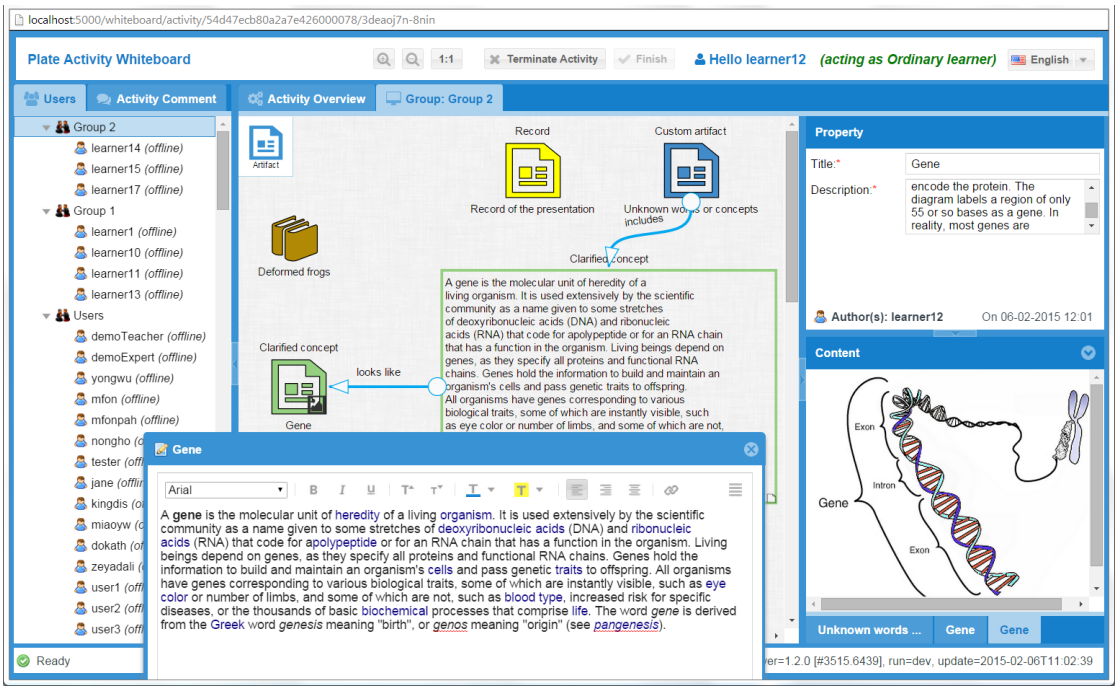


Figure 7.6: A screen-shot of the *PBL Online Whiteboard* (a native PBL runtime player).

essential in the PBL pedagogy; (3) as an important support for PBL educational practice and search work, it was needed as a native PBL runtime player, being specifically implemented to interpret the runnable PBL script instances to carry out the learning activities for each learning participant (corresponding discourse has been elaborated in Part I).

The *PBL Online Whiteboard* was designed to assist learners and/or facilitators in carrying out their collaborative learning task activity by activity according to the learning process definition. When the whiteboard was opened, the current learning activity-associated learning resources, learning tools, expected final artifacts, and other predefined information were already put there.

At the left of the whiteboard, a panel showed all the participants according to the actor organization, current activity, and instantiation arrangement. This panel made the participants aware of their group members and also of other groups. It also showed their status whether online or offline. The chatting session could be established between two participants, within a group, or for a particular learning topic which was bound with a particular learning artifact. A learning space was located on the right side of the whiteboard. This space provided a shared place for participants who were in the same group. The online whiteboard could also become

a private space if a participant was set to individually perform the current learning activity. Similar to the authoring tool, participants could create or edit artifacts on the center learning space to represent their intermediate or final learning outcome.

Continuing the "Deformed frogs" example, as shown in Figure I, the "learner 10" could start the Activity 1.2 "Discuss unknown words or concepts" of the module "Instance 1 of the deformed frogs" from the *PBL Course Management Tool*. Once a participant clicked the link to start the activity, the *PBL Online Whiteboard* opened to carry out the activity for the learning process. In the whiteboard, the participant would then view the given learning resources by clicking the "Deformed frogs" icon, and he or she could freely contribute his or her ideas or knowledge artifacts via adding texts or links, uploading images, audios or videos, embedding external websites, etc. This freedom was also controlled by the whiteboard according to the role operation permission settings fro the participant. For example, if a participant was an *Ordinary learner*, he or she could only modify his own artifacts; if a participant was a *Group leader*, he or she could modify not only his or her own artifacts but also his or her group members' artifacts. Usually, the facilitator had the right to terminate an activity. Once this activity was finished by participants or terminated by their facilitator, the content of the whiteboard would be updated for the next activity based on the process and content control service provided by the PBL course engine. The details about how this control worked was introduced in Section 6.3.

7.4 PBL Language Editor

In Section 6.1.2 we intruded that the IDEE4P had 4 top level function modules. We have also introduced three of these modules. The remaining module is the *PBL Language Editor*. The reason we introduced this tool in the last section of this chapter was that (1) this tool did not provide direct help to facilitate the PBL; (2) it was not crucial for problem-based Learning Design; and (3) the UI of this tool was similar to the UI of the *PBL Authoring Tool*, but less complex than the authoring tool, thus it was easy to introduce it after the authoring tool. However, this did not mean that this tool was not important as the significant of this tool was for PBL ontology research rather than for PBL education practice.

Figure 7.7 shows a screen-shot of the *PBL Scripting Language Editor*. It provided a similar user interface with operation-like functional components to the authoring

tool. The reason they share a similar user interface was that the PBL scripting language and PBL scripts both had the same metametamodel including *Activity*, *Recourse*, *Tool*, and *Artifact*. The *PBL Language Editor* could help PBL experts easily define the concrete syntax of the PBL scripting language, such as defining the phase type, the activity type, the relationship between phases and activities, the properties of an abstract phase (a certain type of phase) and an abstract activity, etc.

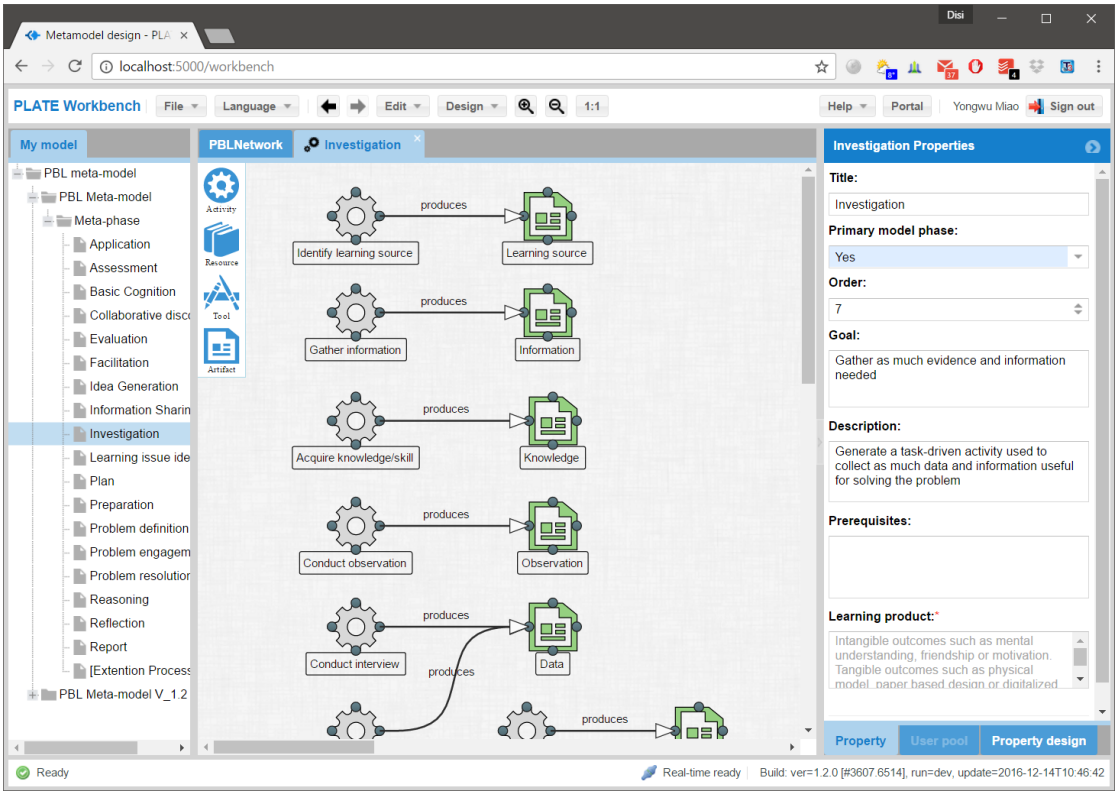


Figure 7.7: A screen-shot of the *PBL Scripting Language Editor*.

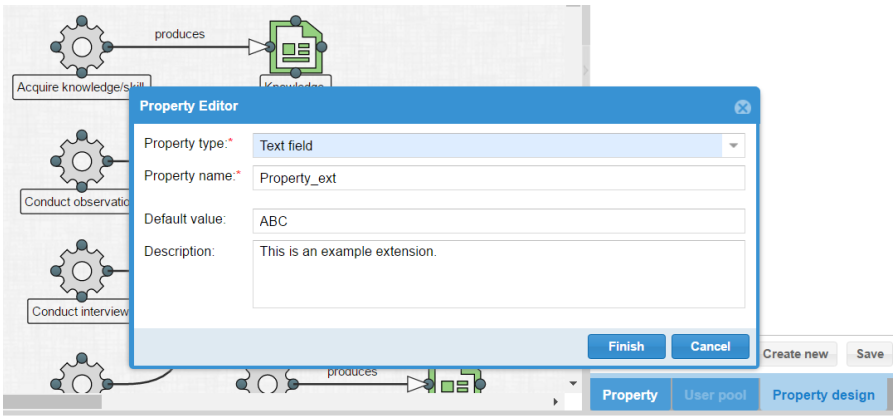


Figure 7.8: Creating property for the definition of the PBL scripting language (a).

For example, the screenshot in Figure 7.7 shows that an expert had defined a list of *Mate-phases*, or phase types. In the phase type *Investigation*, there were a set of activity types defined, such as *Identify learning issue* (which was mentioned in Section 7.1.2 of the phase-activity structure authoring), *Gather information*, *Acquire knowledge/skill*, etc. Each activity type was defined further by associating the expected learning outcome artifact type. About defining the properties, in Figure 7.7, the expert defined the phase type *Investigation* to have the property *Title*, *Primary model phase* or Not, Goal, etc. For example, the *Investigation* was a *Primary model phase* while *Report* was not. However, if the kind of phase was primary or not was not the focus of this thesis.

Figure 7.8 and 7.9 shows the way to create a property for the definition of the PBL scripting language. In this screenshot, when creating a property, we could define the *Property type*, *Property name*, *Default value*, *Description*. Additionally, the value of those elements were the metametatype (type system for metametamodel). For example, from Figure 7.9, the options of the *Property type* included *Single line text* (in the PBL authoring stage it would be a setting input via a Text field), *Multiple-line text* (in the PBL authoring stage it would be a setting input via a Text area field), etc. For thesis space constraints, I do not plan to present much more about the details of the metametatypes and the corresponding options.

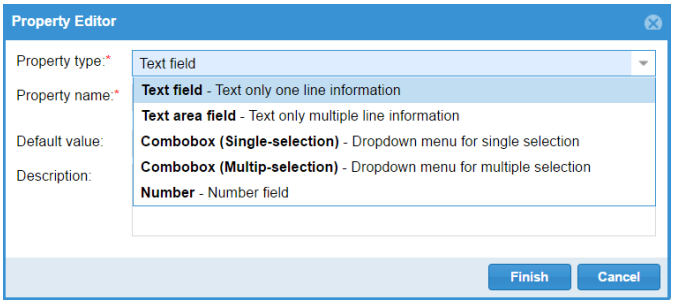


Figure 7.9: Creating property for the definition of the PBL scripting language (b).

In the left-hand side list of Figure 7.7, there were several different versions of PBL scripting language created. This was what was mentioned in Section 5.5.2, that the IDEE4P could, as a PBL research tool, support to define and store different versions of the PBL scripting language. Experts were able to determine which version would be used for problem-based Learning Design. This mechanism could empower PBL experts to test their different considerations and findings from the PBL pedagogy.

Part III

Evaluation

Chapter 8

Usability and Acceptance Evaluation

8.1 Introduction

To investigate the usability, functionality and the acceptance of the modeling concept of the authoring tool, we conducted a pilot study. In this study, 18 students (N = 18, 11 females; age: M = 25.11, SD = 2.88, range: 20 – 30 years) from computer engineering and applied cognitive and media science participated. They were asked to use the authoring tool to represent given textual learning scenarios as a process diagram. The participation time was approximately two hours.

8.2 Design of Study

In this pilot study, the participants were given an introduction into the topics of Learning Design and problem-based learning in form of a short text. After reading the introductory information, they listened to a short demonstration (time duration approx. 15 minutes) of the concept and key functions of the PBL authoring tool. After the demo, every participant received a task description and a manual.

The participants' task was to model two out of three textual learning scenarios using the authoring tool (Appendix A is the textual details of these scenarios). The learning scenarios were taken from the domains of medical education and learning in school, which included a biological scenario: *Deformed Frog*; a physical

scenario: *Getting Attention at an Intersection*; and a medical scenario: *The Heart and Circulatory System*. They differed in length and complexity. These scenarios were given to the participants in a manner of round robin in order to make sure that they were distributed randomly.

After finishing the modeling tasks using the authoring tool, the participants were requested to answer a questionnaire that consisted of textual questions and demographic questions to assess how the participants perceived the interaction with the software. To produce a questionnaire fitting to the research goals, some questions for assessing the system usability were taken from IsoMetrics questionnaire. The participants were requested to state their agreement to the statements referring to usability dimensions of ISO 9241-10 on a 5-point Likert scale. Further questions included specific topics dealing with the authoring tool, including questions about the ease of modeling, the concept of modeling, and how the tool provided modeling. The survey also contained two optional open questions where the participants could state their opinion. For example, such as "Regarding authoring tool, what do you think needs most improvement, and why?" and "Do you have additional comments on the material used in the study or regarding the questionnaire?" The material used in the study was provided in German and English depending on the participants' preference.

8.3 Data Collection

Four aspects of data were collected in this study. Those aspects were separately about the prior knowledge and material difficulty, the usability, the acceptance of the modeling concept and the answers of some open questions. By doing the statistic, the data from participants who did not finished the modeling task or the questionnaire were excluded.

8.3.1 Prior Knowledge and Material Difficulty

The prior knowledge of the participants was assessed at the beginning of the questionnaire using a 5-point Likert scale. The calculation of the arithmetic meant that the participants had a little bit less than average previous knowledge regarding the topics learning design (M = 2.33, SD = 1.24) and problem-based learning (M = 2.50, SD = 1.38) whereas the sample also included individuals who had a

great previous knowledge in at least one of the topics and individuals who had no previous knowledge at all.

The perceived difficulty of modeling the three different scenarios was also included in the analysis. The scores of modeling difficulty differed among the scenarios whereas the medical scenario *The Heart and Circulatory System* was perceived as the most difficult one (M = 3.75, SD = 1.06) and the biological scenario *Deformed Frog* was perceived as the leasts difficult one (M = 2.36, SD = 1.12). The result showed that the textual material used in the study for the modeling task was generally not difficult to understand.

8.3.2 Usability

To assess the usability of the software, the scores for each usability dimension included in the questionnaire were computed. The negative questions were reversed before being included in the descriptive analysis. Table 8.1 shows the resulting overall scores for the dimensions of the suitability for the task, which were self descriptiveness, controllability, conformity with user expectations and learn-ability. Appendix B lists the questionnaires about usability.

Usability			
Dimensions	**Number of Questions**	**Mean**	**Std. D**
Task suitability	7	3.52	0.39
Self descriptiveness	9	2.96	0.52
Controllability	4	3.91	0.22
User expectation conformity	4	3.99	0.33
Learnability	6	3.52	0.52
Intuitivity (Global)	1 (7 Points)	4.78	1.96

Table 8.1: Means and standard deviation of scale items of the usability dimensions.

The scores of task suitability and controllability suggested that the participants considered the software as suitable (suitability for the task, M = 3.52, SD = 0.52) for completing the given tasks (translating a textual scenario description into a graphical representation), and they were relatively satisfied with the options for controlling the system (e.g. possibilities of navigation, controllability, M = 3.91, SD = 0.22). The software also seemed to match with the users' expectations (conformity with user expectations, M = 3.99, SD = 0.33) and the interaction was considered as easy to learn (learn-ability, M = 3.52, SD = 0.52). The self descriptiveness (M = 2.96, SD = 0.52) was rated generally lower than the dimensions

mentioned above. The global item for perceived intuitivity of the system usage reached a higher than average score (M = 4.78, SD = 1.96, range: 1 - 7). However, participants showed a relatively big divergence in this point.

8.3.3 Acceptance of the Modeling Concept

Further sections of the questionnaire deal with the ease of modeling different structures in the course plan. Table 8.2 shows the results. The participants were requested to state their agreement to various statements referring to the ease of modeling using a 5-point Likert scale. The results suggested that it was perceived as quite easy to model the user organization (M = 4.22, SD = 0.73) and the overall process of the course plan (M = 4.39, SD = 0.85). Also the navigation between the different layers did not seem to cause any problems (M = 4.00, SD = 1.14). The results for the perceived ease of modeling the single phases of the course plan indicated that this part was perceived as not very easy but still not difficult (M = 3.00, SD = 1.09).

Modeling		
Item (It was easy to ...)	**Mean**	**Std. D**
... model the user organization.	4.22	0.73
... model the overall process for the course plan.	4.39	0.85
... model the single phases of the course plan.	3.00	1.09
... get to the model of the single phases starting at the model of the overall process.	4.00	1.14
... select adequate phase types for modeling the overall process.	2.94	1.26

Table 8.2: Ease of modeling different structures of the course plan.

Moreover, the participants were requested to state their agreement to statements referring to the ease of choosing types for elements when modeling the single phases of a course plan. The results in Table 8.3 indicated that it was perceived as relatively difficult to select an adequate type when creating a new activity element. The perceived ease was M = 2.94 (SD = 1.11), which was a bit low compared to the other scores, e.g. for selection of artifact types (M = 3.44, SD = 1.46).

Table 8.4 showed that the separation of the overall process model and the single phases seems to largely support the expressiveness clarity of the overall script (M = 4.28, SD = 0.90) in comparison to the textual scripts given in the evaluation tasks. On the contrary, the fact that the separation facilitated the finding of

Modeling		
Item (When I was modeling the single phases, it was easy to ...)	**Mean**	**Std. D**
... select the activity types.	2.94	1.11
... select the resource types.	3.40	1.50
... select the tool types.	3.71	1.59
... select the artifact types.	3.44	1.46
... select the action mode (type of collaboration) for activities.	3.50	1.10

Table 8.3: Ease of types choosing in modeling.

relations between single phases was not as clear to follow in comparison to the concept of separating the overall process. However, it still reached a score higher than average (M = 3.56, SD = 1.15).

Modeling		
Item (The separation of the overall process model and the single phase models ...)	**Mean**	**Std. D**
... supports the expressiveness clarity of the overall script.	4.28	0.90
... facilitates finding relations between the single phases.	3.56	1.15

Table 8.4: Agreement of separating the overall process model into different layers.

We also asked some questions related to the positioning of elements and the modeling of a sequence using the options provided by the software (using "top-down" spatial order or explicitly using arrows). The results showed that most of the users used arrows for modeling process sequences between activities (M = 4.56, SD = 0.62). In contrast, using the spatial order to define a sequence of elements only reached a score of M = 3.67 (SD = 1.41). The overall ease of modeling sequences was M = 3.56 (SD = 1.1). Figure 8.5 shows the results.

8.3.4 Open Questions

The optional question, "Regarding the authoring tool, what do you think needs most improvement, and why?" provided additional potentials for improvement. The field was used by seventeen of the eighteen participants to note their opinion. The entries covered suggestions on the functional level as well as such referring to the modeling or errors that occurred while using the software. The result is

Modeling		
Item	**Mean**	**Std. D**
It was easy to model process sequences for the phases.	3.56	1.10
For modeling process sequences for the phases I used the spatial order ("from top to bottom").	3.67	1.41
For modeling process sequences for the phases I used arrows between activities	4.56	0.62

Table 8.5: Feedback referring to defining the process sequence using different methods.

summarized in Table 8.6.

Improvement Suggestions		
Aspect	**Suggestion / Statement**	**Frequency of Mentioning**
Functional	Shortcut support (copy, paste, delete)	5
	Multi-item copy	1
	Possibility of changing fixed properties such as activity types	7
Modeling	Order of activity types (selection dialog) not clear	4
	Connecting activity elements not intuitive (dashed line)	2
Assistance	Demand for additional support while using the software (tooltips, pop-ups, mouse-overs etc.)	9

Table 8.6: Summarized results of the optional questions about the improvement suggestion of the PBL authoring tool.

On the functional level, many entries contained the wish for shortcut support, especially for copy, paste, and delete. A function for multi-item copy or cloning of structures was also wished for by one of the participants. Also the subsequent possibility of changing activity and phase types was mentioned several times (n = 4).

Regarding the modeling, some of the comments (n = 4) stated that the order of activity types in the selection dialog was not clear. Some participants reported that they felt confused that the options position in the dialog list changed depending on the chosen phase type. Some of them suggested an alphabetical order or the segmentation into semantic sections to facilitate the retrieval and selection of activity types. Being forced to use the dashed line to connect activities in the phase layer also seemed to be unintuitive for some of the users. One participant

noted that he did not know what to do if he intended to visualize a sequential process of various dependent activities.

Many of the optional comments contained the demand of additional assistance while working with the system (n = 9). As suggestions, the participants named pop-ups and tool-tips with hints or helping explanations (e.g. Why is the connection of two specific elements not allowed?), especially for explanations referring to the activity types. One suggestion was to add short explanations of activity and phase types that appeared when moving the mouse over the corresponding list entry.

8.4 Activity Sequence Analysis

In addition to the questionnaire-based statistic study, we also tracked the actions of each participant when they were completing the tasks. This action tracking proved that there was a good usability and the good acceptance of the modeling concept provided by the authoring tool from another aspect.

8.4.1 Supporting Flexible Modeling Style

General speaking, from the activity data we knew that, most participants represented the learning scripts using a gradually-refine style. They first define the phase, and then defined all activities for the phase. When they defining an activity, if a tool or an artifact needed to be associated, they usually would make the association immediately. However, the activity structures also showed two modeling style variations. One was the way they defined the phases; the other was the time they made the actor-organization model.

Regarding the first variation, some participants first defined all phases and then defined activities for each phase, while others defined one phase and the phase's activities, then they came back to define the next phase and its activities again. Regarding the second variation, some participants modeled the entire actor-organization at the very beginning while some gradually completed it during phase-activity modeling. This difference largely depended on the way that how the textual script described the learning scenario. It the scenario provided a relative more clear structure, participants tended to define all phases first before they began to define the

activities for the phases. Or if the textual script introduced the actor-organization characteristic at the beginning, then participants modeled it also at the very beginning. This finding showed that the authoring tool was able to flexibly adapt the different modeling styles according to different learning scenario descriptions.

8.4.2 Easy to Learning Modeling Concept

Additionally, another important finding was that the modeling speed of the participants could largely increase in a short time. Figure 8.1 was an action sequence of one participant as an example. Those actions were automatically tracked by the authoring tool. From this diagram we could see that this participant used approximately 64 minutes to model the first learning scenario (physical scenario: *Getting Attention at an Intersection*), and took about 46 minutes to model the second scenario (biological scenario: *Deformed Frogs*). This diagram shows at what time the people performed what action. For instance, at 8:22 am, this participant created a phase element and then moved it to an appropriate position. Table 8.7 showed the time consumption of each participant when they model each scenario.

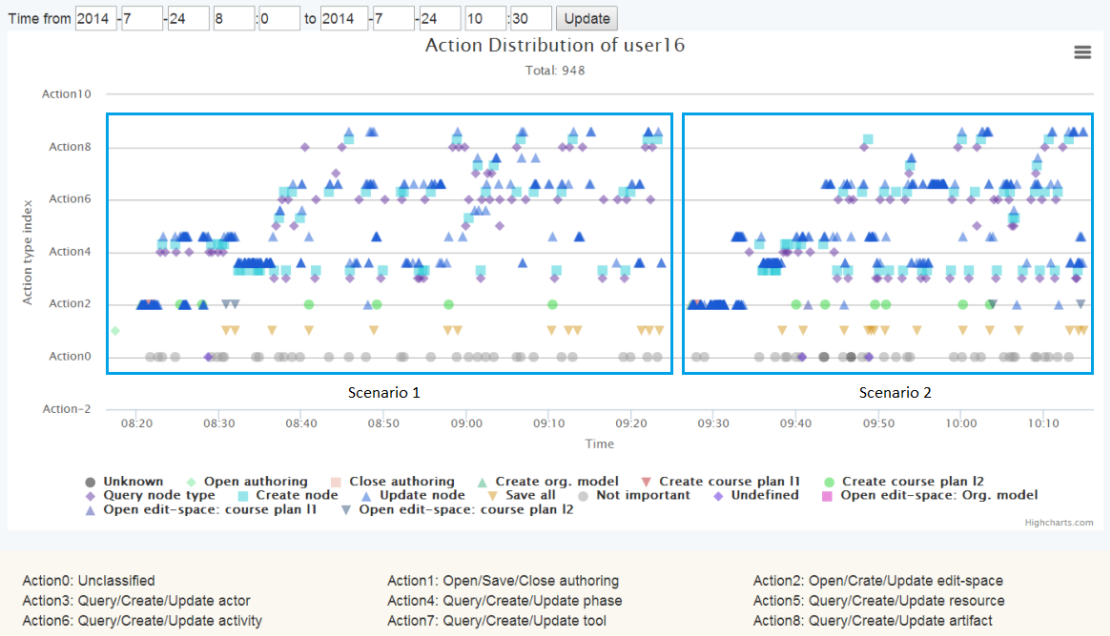


Figure 8.1: A sample of the sequence of authoring action of a participant.

If we observed the difference of the time consumption of the same scenario, between it being as the first modeling task and it being completed as the second modeling task, we found that, for the same scenario, if it was arranged in the second modeling

	Scenario 1		Scenario 2	
User name	Scenario	Time consumption	Scenario	Time consumption
user01	Med.	58 Min.	Bio.	38 Min.
user02	Bio.	48 Min.	Med.	26 Min.
user03	Phy.	57 Min.	Bio.	52 Min.
user04	Bio.	42 Min.	Phy.	40 Min.
user05	Med.	44 Min.	Phy.	33 Min.
user06	Phy.	60 Min.	Med.	30 Min.
user07	Bio.	36 Min.	Phy.	35 Min.
user08	Med.	60 Min.	Phy.	24 Min.
user09	Med.	35 Min.	Bio.	45 Min.
user10	Phy.	40 Min.	Bio.	35 Min.
user11	Phy.	52 Min.	Med.	21 Min.
user12	Bio.	55 Min.	Med.	30 Min.
user13	Bio.	55 Min.	Phy.	20 Min.
user14	Med.	50 Min.	Phy.	22 Min.
user15	Med.	53 Min.	Bio.	25 Min.
user16	Phy.	64 Min.	Med.	46 Min.
user17	Phy.	46 Min.	Bio.	56 Min.
user18	Bio.	59 Min.	Med.	34 Min.

Table 8.7: Time consumption when modeling each scenario of each participant (Med. = Medical scenario; Bio. = Biological scenario; Phy. = Physical scenario; Min.= Minute).

task, the modeling time was largely reduced. For instance, modeling the medical scenario took 50 minutes on average (M = 50.0, SD = 9.32) when it was arranged as the first modeling task, while it just took 31.2 minutes in average (M = 31.2, SD = 8.50) when it was arranged as the second modeling task. More details of the statistical results shown in Table 8.8. Figure 8.2 is the corresponding diagram of the statistical results shown in Table 8.8.

	As the 1st modeling task			As the 2nd modeling task		
	Med.	Bio.	Phy.	Med.	Bio.	Phy.
	58 Min.	48 Min.	57 Min.	26 Min.	38 Min.	40 Min.
	44 Min.	42 Min.	60 Min.	30 Min.	52 Min.	33 Min.
	60 Min.	36 Min.	40 Min.	21 Min.	45 Min.	35 Min.
	35 Min.	55 Min.	52 Min.	30 Min.	35 Min.	24 Min.
	50 Min.	55 Min.	64 Min.	46 Min.	25 Min.	20 Min.
	53 Min.	59 Min.	46 Min.	34 Min.	56 Min.	22 Min.
Mean	50.0	49.2	53.2	31.2	41.8	29.0
Std. D	9.32	8.84	9.00	8.50	11.48	8.10

Table 8.8: Time consumption when modeling each scenario of each participant (Med. = Medical scenario; Bio. = Biological scenario; Phy. = Physical scenario; Min.= Minute).

According to the design of this evaluation, each participant modeled different sce-

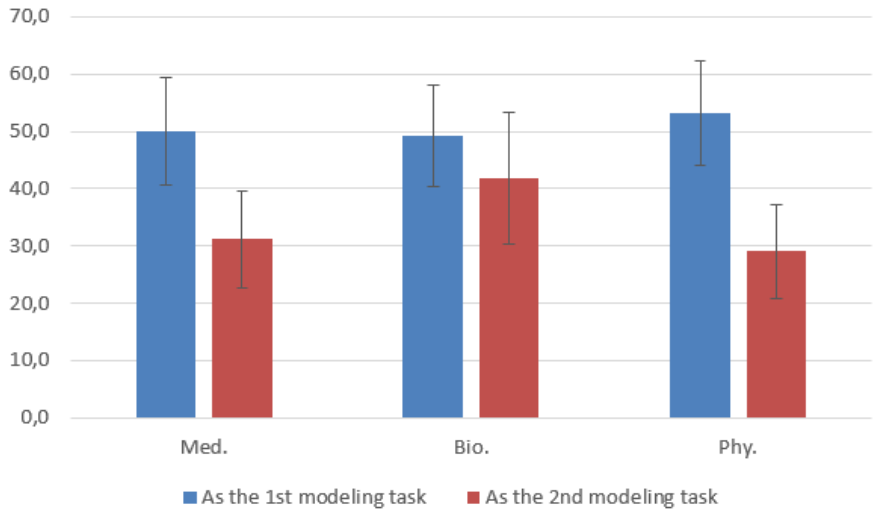


Figure 8.2: Rapidly reduced time consumption of PBL scenario modeling when using the PBL authoring tool.

narios, which meant the reduction of time consumption was not because the same people modeled the same scenario twice. The time reduction was mainly because people rapidly became familiar with the modeling concept and the authoring tool after modeling the first scenario.

8.5 Results and Discussion

The results indicated that although modeling the problem-based learning scenarios was not a familiar task for most of the participants, the tool provided a possibility to let them relatively easily achieve the task requirement. Either from the software usability viewpoint or from the viewpoint of accepting the authoring way for the scenario modeling, most participants felt it was easy to learn to use the tool and that it felt very natural to understand how to use the diagram-based user interface to transfer the meaning of different forms of textual-based learning scripts to digital forms of learning models. This authoring tool made them realize that it was possible to design problem-based learning as a more intuitive representation that was easily to understand, was able to reuse, provided better communication, had sharable manner, and meanwhile was able to support technology enhanced online or blended problem-based learning.

However, this evaluation also revealed issues of this authoring tool. Regarding the usability and the functionality, additional functions for support and assistance

needed to be added, and better explanations about modeling elements and activity types needed to be provided. Regarding the modeling concept provided by the tool, the results of the questionnaire analysis indicated that modeling a sequence with the options provided by the software was sometimes not fully clear or difficult to realize for the users (M = 3.56, SD = 1.1).

About the usability and the functionality, if the user did something wrong or unintended (e.g. trying to connect nodes that could not be connected), the software needed to provide helpful feedback why the current action was not allowed (e.g. a short explanation in form of a tool-tip or pop-up dialog). Such an improvement of user support could also increase the relatively low level of self-descriptiveness. Additionally, a function for previous defined cloning structures and short-key support needed to be added to make working with the authoring tool faster and easier.

About the modeling concept, the sequential modeling of events taking place during certain phases of the course plan especially seemed to cause problems. The results hint that it might not be fully clear when to use the spatial order and when to use edges between elements to show their dependency and processing sequence. The open question referring to errors that occurred while using the software also revealed this problem and that it was often not clear why links between certain objects (e.g. actor and artifact) were not possible. This led to the assumption that the "translation" of a textual scenario into a graphical representation regarding the identification and visualization of dependencies and sequences of objects using the given modeling elements seemed to be quite difficult. Another problem was the difficulty of choosing activity types in the selection dialog when setting up a new activity element in the phase layer. To facilitate this process, short explanations to each list entry could be added. Another option would be to clarify the order of the entries among the list (e.g. by adding semantic headlines).

Additionally, we also received some controversial improvement suggestions. Some participants suggested that it would be better to arrange the type option list of phase or activity or other elements in alphabetic order, while the system design experts argued that it was better to order them according to the usage frequency and categorized based on their underneath relationship. The reason why the participants wanted it in alphabetic order was that they mainly just needed to transfer the words of the task scenarios to the tool. Thus they subconsciously wanted to "find" the words closest to the words in the textual description. The finding action needed an alphabetic order list. However, the tool was not designed just for this

kind of usage scenario; it was also designed to help people who had only abstract mental learning scenarios in mind to represent their ideas in an appropriate way. In this case, the authoring tool needed to provide a guidance of the type selection. Therefore, making the recommended types on the top of the option list and grouping similar types together was more reasonable, which usually meant we needed to break the alphabetic order.

Chapter 9

Evaluation in Teacher Training

This section is partially based on the publication "Using a PBL Authoring Tool to Train Teachers in Designing an Online PBL Unit" by Miao, Y., Samaka, M., Wang, D., Ali, Z., and Romanowski, M. (Miao et al., 2015b).

9.1 Introduction

Chapter 8 presented a formative evaluation of the partial functions of the IDEE4P system. In comparison, this chapter will present a summative evaluation.

In order to investigate whether the PBL authoring tool could be used to train teachers in designing an on-line PBL unit, we conducted a pilot study. We adopted an approach of learning by design to educate novices to acquire PBL knowledge and become familiar with the procedure and informed decisions. Here, we used the approach "learning by design" defined by Koehler and Mishra (Koehler and Mishra, 2005) because their focus, like ours, was on teacher learning and professional development.

The pilot study was conducted in the College of Education at Qatar University. Participants in the pilot study were students from the Masters in Education program and were enrolled in the end of the program internship having already completed an advanced curriculum development and design course. Most of the participants were still working as teachers in primary, preparatory and secondary schools or working in the education-relevant fields. This pilot study was arranged as a part of the course. In the course, two sessions were arranged and each ses-

sion took three hours. In the first session, participants were introduced to PBL, including basic PBL concepts, principles, and benefits. They were instructed how to design an on-line PBL course, in particular, to choose ill-structured problems, to design various process structures and to arrange individual and collaborative activities with various communicative and collaborative tools. At the end of the first session, participants were briefly introduced to the PBL scripting language and the PBL authoring tool. Participants were required to create a user account in the PBL Workbench and to learn the tool by using a user manual and a tutorial video on their own. In the first half of the second session, participants were guided to represent a pre-designed PBL course with the PBL authoring tool step by step. In the second half of the session, participants applied what they learned through continually working on the representation of the pre-defined PBL course. During this time, some participants asked questions that were answered immediately in the class. Then the participants had to complete an assignment within ten days to create a PBL script with the tool. The assignment was centered on their authentic design problems. All participants (N = 17) completed their PBL scripts on time. Finally, participants were required to response a questionnaire. Seventeen responses were collected and all were valid responses.

9.2 Data Collection

Two types of data were collected. The first type was from participants' responses to the questionnaire, the second type of data were the PBL scripts created by the participants using the PBL authoring tool and stored in the database persistently.

For the first type, the questionnaire consisted of five sections:

- **Section I** asked questions about the participants' background;
- **Section II** addressed the computer literacy of the participants;
- **Section III** contained Likert-scale questions (selecting one of five responses ranging from 1: strongly disagree to 5: strongly agree) designed for collecting the participants' attitude toward the PBL authoring tool;
- **Section IV** included open questions designed for collecting the participants' feedback.

The questions in Section III were designed by referring to the Technology Acceptance Model (TAM) proposed by Fred Davis (Davis, 1989) as a means of predicting

technology usage. Because it was the first time for the participants to design online PBL units, they might not have been able to judge the effectiveness and efficiency of use of the tool. The questions were mainly relevant to easily use and learn. Note that several reversed items were designed in Section III where the participant responded to the opposite of a question. For example, in the questionnaire, an original item was "it is difficult to use and to learn a two-layer process structure of a PBL script." It was readily reversed to state "the two-layer process structure is easy to use and to learn." The responses for the 'reversed' items were then reversed before they were scored. In Table 9.2, we used a phrase "the two-layer process structure" to represent this question.

According to participants' responses to the questions in Section I, it was evident that all participants were university students and most of them were also teachers in subject areas including Arabic, math, English, health, and science. Table 9.1 shows the answers of the students to the questions about background information. In Table 9.1, Question A was about the prior knowledge of PBL from the participants. The data showed that most of the participants had minimal prior knowledge. The questions from B to F were concerned with computer literacy. In detail, these questions were:

- Question A: "To what extent do you know problem-based learning before attending this workshop?".
- Question B: "Are you skilled at using generic computer tools such as Word or PowerPoint?".
- Question C: "Are you skilled at using communication tools such as chat rooms, discussion forums, blogs, or wikis?".
- Question D: "Are you skilled at using learning management systems such as Moodle or Blackboard?".
- Question E: "Are you skilled at using education-specific tools such as digitalized whiteboards, educational simulators, or online questionnaire authoring and responding tools?".
- Question F: "Are you skilled at process modelling with UML or a computer programming language?".

Because the different skills had different influences in using the PBL authoring tool, we calculated the total computer literacy, noted as G, of a student by using

a weighted sum in a way

$$G = SUM(B * 1 + C * 2 + D * 3 + E * 4 + F * 5)/15$$

where G shows that the levels of computer literacy of most participants were around an average level.

For the second type, the collected PBL scripts were assessed using three scoring rubrics: completeness, contextualization, and reasonableness. Each rubric score was ranged from 1 to 10 and the grade of a PBL script were the mean of three rubric scores. Two experts rated the students' PBL scripts and each final score was means of scores given by the two experts. The final scores were shown in row H of Table 9.1.

Student	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**Mean**	**Std. D**
Question A	4	3	5	3	3	3	4	4	1	4	2	2	1	3	4	5	5	3.3	5.05
Question B	4	5	5	5	4	5	5	5	5	4	4	4	5	5	5	5	5	4.7	0.47
Question C	4	3	3	4	3	4	5	5	2	4	4	2	3	5	3	5	5	3.8	1.03
Question D	4	4	4	4	4	5	4	5	2	4	5	4	4	4	4	5	5	4.2	0.73
Question E	3	4	3	3	2	4	4	4	2	5	3	4	4	4	5	5	3	3.6	0.93
Question F	2	1	3	1	1	2	1	2	1	3	3	1	1	1	1	2	3	1.7	0.85
Computer literacy (G)	3.0	2.9	3.3	2.8	2.3	3.6	3.2	3.7	1.8	3.9	3.6	2.7	2.9	3.2	3.2	4	3.8		
Score of script (H)	6	9.5	9.2	9.5	10	8.7	8.5	8.2	8	8.2	8	7.7	7.7	8.5	10	8.5	9.2		

Table 9.1: Students' prior knowledge in PBL and their computer literacy.

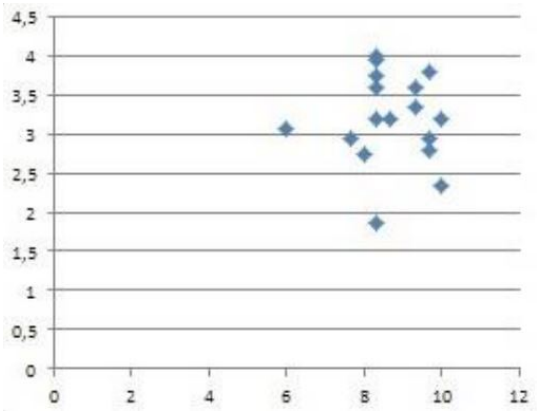
9.3 Results

This subsection presents the results from the data analysis and discusses relevant issues. After a short training, students worked for ten days and then created their own PBL scripts. Each student received a final score as shown above. We analyzed the relations between the scores of the students and their prior PBL knowledge and their computer literacy. As depicted in Figure 9.1a, we found that the influence of computer literacy on the final score was not significant. However, as shown in the Figure 9.1b, prior PBL knowledge had a slight positive influence on the final score, but one was obviously deviated.

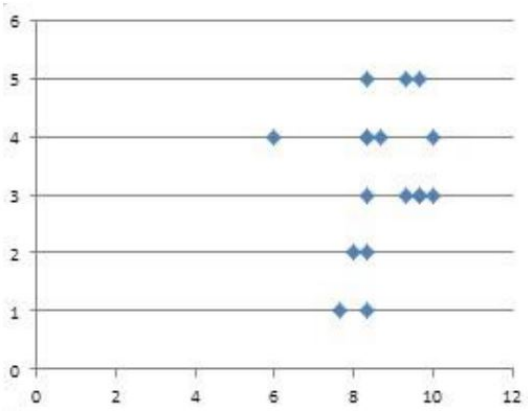
Table 9.2 shows all scale items and the basic statistical data of the responses. The means of the scores of all fifteen items in Section III were larger than 3.0, and the total mean of the scores was 3.6. Most participants thought it was easy to learn

Items of Questionnaire Section III	Mean	Std. Deviation
3.1. the two-layer process structure	3.4	1.00
3.2. the vocabularies selected in the language	3.9	0.97
3.3. the activity structure that includes actors, artifacts and their relations	3.5	1.01
3.4. the properties of PBL script, phase, activity, and artifact	3.6	0.79
3.5. the translation from a narrative into a diagram-based script	3.2	0.90
3.6. the organization structure	3.9	0.90
3.7. the assignment of an activity to an actor	3.8	1.09
3.8. the choice of a work mode	3.7	0.99
3.9. the choice of a collaboration mode	3.2	1.19
3.10. the choice of synchronous/asynchronous communication	3.3	0.99
3.11. the choice of tools	3.7	0.77
3.12. the choice of a learning setting with certain devices	3.5	1.07
3.13. the representation of an artifact transferring	3.6	0.80
3.14. the choice of completion condition	3.4	0.80
3.15. the estimated duration	3.6	1.33
Mean of total	3.6	0.97

Table 9.2: Means and standard deviation of scale items on the ease of use from Section III of the questionnaire: about the PBL scripting language and the PBL authoring tool.



(a) The relation between levels of the computer literacy (G, ranged from 0 to 5) and scores (H, ranged from 0 to 10).



(b) The relation between the prior knowledge (Question A, ranged from 0 to 5) of PBL and scores (H, ranged from 0 to 10).

Figure 9.1: The relationship between students' computer literacy, prior knowledge and scores of their scripts.

to make informed decisions, and the tool was easy to use to design an on-line PBL unit.

About the feedback of the open questions, in relation to the usefulness of the tool in designing an on-line PBL unit, students wrote: “I used the PBL Workbench for a science lesson. It was suitable for the topic.” “When I used the PBL Workbench I did not have any difficulties performing a task. There were various possibilities to work with.” “I liked the way. It allows connections to be made between various elements, actors, activities, etc. I also liked that it provides clarity to every phase and activity as it asks for goals and other details.” “This tool was amazing in helping me develop the plan of how to conduct performance management at the school especially with the complications of connections to be made.” “It was new and exciting experience for me”.

In response to the question about whether it was easy to learn, students stated: “I would not say easy. It took me time to understand the thing I needed to represent my design and not sure if it’s the right thing. Yet I think with more understanding and practice it could become easy in time and the use of different design form.” “First I thought it was difficult. When I started to work with it, I found it is not that difficult, yet not an easy one to use.” “It will be easy if there are PBL model templates to help teacher to design one because it took me a long time to design a course plan.”

When answering the questions regarding the vocabularies and rules provided by the PBL scripting language, the participants reported that the vocabularies and rules were very helpful. They stated: “I think that a pre-defined list of choices facilitate the design. It saves time and efforts.” “Of course, it facilitates the work, especially for the teacher with little experience, while for expertise teachers it might limit the options, or ideas.” “I did not understand these terms until developing the course plan.” “I think they are ok and there are a lot of options to choose from which represent the basic items that are used in phase or activity. Yet I think if we could have the possibility to add our own ones.” “From my point of view it enhances and facilitates process. It gives us an option to explore and think critically. I found it appropriate, and enough to help in PBL. On the other side, yes more options will be good addition, as ‘more sugar, more sweet.’”

In terms of aspects of the tool that were most valued, students wrote: “It provides students with greater flexibility in developing solution strategies.” “It is flexible as we are able to define as many phases as we wish and as many activities as we wish within each phase I used in my design three phases and more than 2 activity in each and it was easy.” “PBL Workbench is very flexible and helpful, easy to use,

and has many multiple options that represent various forms." "The workbench is flexible as it does not restrict us to any particular PBL model and gives room to design a PBL with as many phases we require depending upon the project." It seemed that flexibility was an important feature of the tool.

9.4 Discussion

Although the tool generally satisfied the need of designing on-line PBL unit, the data also revealed that students needed more instruction and assistance. Some students noted that: "Lots and lots of options and information which I need more training on." "I did develop my lesson using all of these things, it is not difficult, yet I think if there is a way to explain them more or demonstrate the different way they can be represented." "I took a long time to develop PBL course plan. I had to log off and log on for several times because I become tired and keeping track of developing plan. During this, I thought it could be easier if there were ready-made PBL example models the teacher can choose and change what is needed for the lessons. For example, it became similar to Microsoft publisher that have ready-made template to use and change."

To improve the tool, students suggested: "Editing is a challenge. If a mistake takes place, the whole process, phase, actor had to be redone from scratch with all the fields required which is very difficult. Editing would make it much easier to modify." "If the tool could provide its own resources that are presented in accordance with the standard used in schools one can use to save time." These were good suggestions and we developed functions to meet these requirements.

Feedback received from the participants indicated that most agreed that the PBL authoring tool was easy to learn and use for designing an on-line PBL unit. The tool provided guidance to make informed decisions and provided options for choosing. Participants especially emphasized that vocabularies and rules specified in the PBL scripting language made it easy to understand and design a PBL script and that the tool provided flexibility. The responses to the questions about students' perception of the tool were basically positive, and most PBL scripts created by the students were quite good. These results were encouraging since the participants in the pilot study had little prior PBL knowledge and only had average levels of computer literacy.

Based on the feedback from the participants, it was important to instruct potential users on the theory of PBL and explain what were informed decisions and possible choices. Also, it was important to provide more examples and help on how to use the PBL authoring tool and to provide support to users as they developed an on-line PBL unit. The feedback from participants in this pilot study indicates that the PBL authoring tool would be useful for teachers to develop an on-line PBL unit. This pilot study was conducted as a formative assessment. According to the feedback, we improved the PBL scripting language and the PBL authoring tool.

Part IV

Conclusion

Chapter 10

Summary and Outlook

10.1 Summary

All of our work is based on the large background of the demand of the 21st century competencies. Therefore, our vision is to satisfy this demand to some extent. In order to achieve this, we need to make ourselves clear about the content of the competencies. However, different organizations or groups develop different definitions of competencies. Nevertheless, we quickly found that the major content is similar. Generally speaking, the competencies include *critical thinking*, *information literacy*, *reasoning and argumentation*, *innovation*, *flexibility*, *initiative*, *appreciation for diversity*, *metacognition*, *communication*, *collaboration*, *responsibility*, and *conflict resolution*.

One important pedagogical approach that can meet the competencies demand above to a certain degree, and an innovative education approach, Problem-based Learning (PBL), has emerged. Although this approach was initially developed for the medical education, with the continuous expansion of the study scope and the innovative extension of the boundaries, more and more scholars find that problem-based learning not only suits the medical education, but also for other professional areas. Unlike the discipline-centered curricula where the teacher is the center, and rather than providing a predefined solution a so-called right one, in PBL realistic case problems are used as stimuli and the solutions expected from students are multiple. This approach drives students to actively and collaboratively gain knowledge and develop their critical thinking, problem-solving, team-work, and self-directed learning skills through a staged sequence of problems that are pre-

sented in context, together with associated learning materials and support from teachers.

Nevertheless, despite that PBL has so many advantages and benefits and is widely recognized, it is not widely applied yet. One important reason for this kind of divergence phenomenon is that an appropriate implementation of PBL is not easy for current teachers. The reasons for this situation are multi-faceted, including that the pedagogy theory is not that easy for ordinary teachers to master, there is no universally agreed set of practices to guide the implementation of PBL, and there is no good enough PBL application to technically reduce the implementation difficulty.

Facing all the findings above, a project named Problem-based Learning Authoring and Transformation Environment, shortly *PLATE*, is created. This project aims at and achieves developing an innovative approach and an online system that helps teachers easily, flexibly, and responsively implement a wide range of models of online or blended PBL, as well as help teachers to make the implicit PBL design process explicit so that the Learning Design quality can be easily improved and represent and enable a traditionally informal description as a formal model used for scaffolding and orchestrating PBL processes. Simply speaking, we need to reduce the difficulty of PBL implementation and help increase the quality of problem-based Learning Design.

In order to achieve this goal, we needed to (1) develop a high-level scripting language and a new generation of learning design environment that flexibly and efficiently supports the development of PBL-infused modules and courses; (2) create a run-time environment that supports the execution of PBL models and scaffolds and orchestrates PBL activities to help students to transit into new roles and to learn within PBL contexts. Consequently, form my perspective, three tools needed to be developed: a *PBL language editing tool* helping PBL experts to develop a high-level scripting language, a new generation *PBL-specific learning design tool* helping teachers make the implicit PBL design process explicit, and a *PBL run-time tool* helping students to transit into new roles and to learn within PBL contexts.

Designing and developing these three tools in a systematic way was a big challenge. Therefore, I engaged an in-depth theoretical research. The research includes summarizing the characteristics of PBL, comparing different models and fundamental implementation principle of PBL, and analyzing the concepts and the general

ideas behind the Learning Design and IMS Learning Design. Accordingly, I also reviewed state of art technology, tools, and systems that are able to facilitate PBL. The tools include content-oriented tools and activity-oriented tools, where is activity-oriented tools are the review focus. Those tools include LAMS, MOT+LD, CopperCore, CopperAuthor, Reload tools, ASK-LDT, COSMOS, eLive LD-Suite, SLED, Edubox, STELLAR, ePBL and etc.

I recognize that those tools either (1) are designed to rely on a certain specific PBL model that may be useful only in certain circumstances and may be not applicable to other situations or deal with the lack of flexibility for teachers to customize PBL models in order to satisfy their individual PBL strategies: currently teachers and students must work by following the workflow implemented in the tools; or (2) the notations provided by those tools are the general concepts of learning activities rather than those specific terms frequently used in PBL practice. It is still difficult to develop PBL courses for teachers. Besides, based on the theoretical research about PBL, Learning Design, IMS-LD, the author further recognize that applying a PBL domain-specific language technology within a model-driven approach is the right direction to overcome the two aspects of shortcomings above.

As a fundamental work of the model-driven approach, a PBL scripting language was proposed by Miao et al. The PBL scripting language is developed based on the concepts of existing CSCL scripting languages, combined with the domain concepts of the PBL after analyzing the features of all existing mainstream PBL models concerning supporting a general expressiveness for representing various forms of PBL models. However, this language idea is more or less just a conceptual guidance which is far less than enough for the technical design and implementation.

Considering that the PBL scripting language is a kind of domain specific modeling language, and the domain-specific modeling as a software engineering methodology for designing and developing systems was developed mainly for industrial production purposes for a long time and tends to be mature, I further analyze the industrial model-driven approach, and then map the essence of the approach to our problem-based learning design context; this is for the purpose of gaining a concrete technical design and implementation methodology for the PLATE system.

I argue that the Learning Design is similar to the model-driven development (or model-driven engineering) because the Learning Design and the model-driven development has the same intention and same working principles. Furthermore, as the theory infrastructure of the model-driven development, the model-driven ar-

chitecture is discussed, especially about how to use the PBL scripting language in the model-driven architecture in the analysis. According to the analysis, the PBL scripting language is the metamodel, the PBL process is the model, and the PBL course is the instance.

Accordingly, the *PBL language editing tool* shall provide the function to edit the metametamodel, helping PBL experts to develop the PBL scripting language as the metamodel; the *PBL-specific learning design tool* shall provide the function to edit the metamodel, helping teachers make the implicit PBL design process explicit as the model; and the *PBL run-time tool* shall be able to interpret the model and carry out PBL courses as the instance. Moreover, there are other mappings: the PBL model can further have two aspects: a PBL platform independent model (PBL-PIM) and a PBL platform specific model (PBL-PSM). The difference between these aspects is the viewpoint. PBL-PIM is for the multiple runtime transformation and the model storage usage, and PBL-PSM is the transformation result of PBL-PIM. Last but not the least, since the PBL scripting language becomes stable and mature only when the tools are developed, I encounter a mutual recursion problem, which means we need to develop a system that will be built upon a fundamental specification, where the specification will be developed and investigated back through the to be developed system. Also based on the analysis of model-driven development, I adopt an iterative engineering approach to solve this problem.

In the implementation work, I utilize the latest emerged ICTs, such as the event driven non-blocking I/O and NoSQL technologies, etc. In comparison, existing applications store Learning Design artifacts by using either traditional relational databases or directly as XML files. Although relational databases are good at data storing and querying, they can not manage this kind of semi-structured data well and are not flexible enough in this system since they are relational and not schema free. The XML file is a kind of ideal media for storing this type of semi-structured data. One drawback is that it is not ideal for the manipulation, such as partial updates, searches, and sub-document sharing. Although there are some combination solutions to manage XML documents through relational database, XML queries are still inefficient (Shanmugasundaram et al., 1999).

As a result, the model-driven approach and the emerging ITCs summarized above significantly help system design and development and I successfully develop four tools and a PBL runtime engine. They are together named as **Integrated Design**

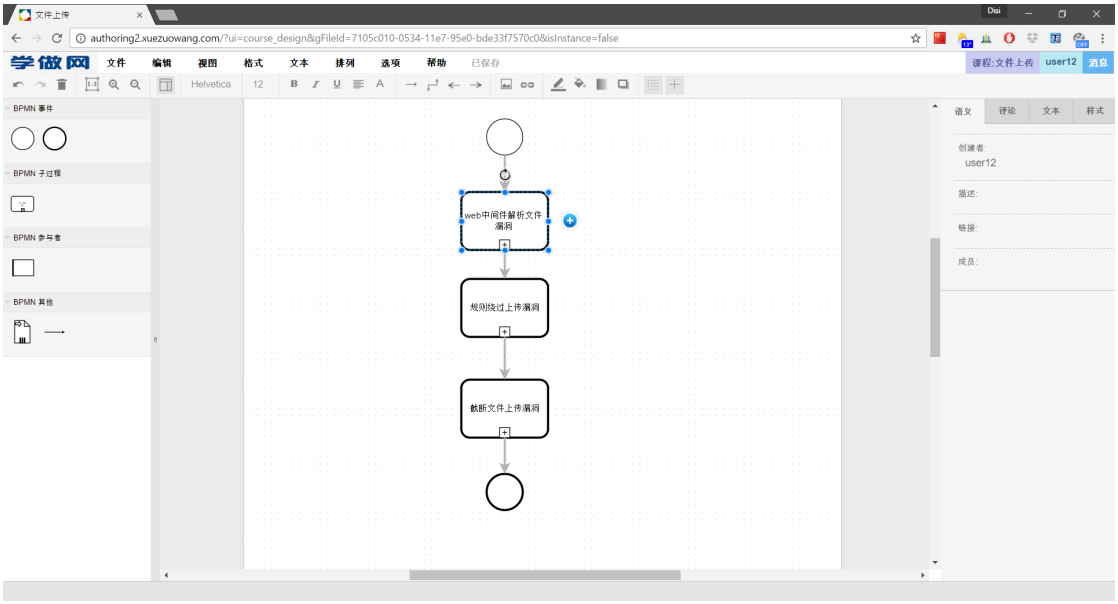
and Execution Environment for PBL, shortly **IDEE4P**. I present two pilot studies that show the usability of the authoring tool, the acceptance of the course modeling concept of the IDEE4P, and the possibility for training teachers through the IDEE4P. From the results of these studies, I conclude that applying a model-driven approach with state of art technologies to enhance the online PBL appears to be a promising effective and efficient support for the research and practice in facilitating the problem-based Learning Design and the corresponding learning implementation.

10.2 Outlook

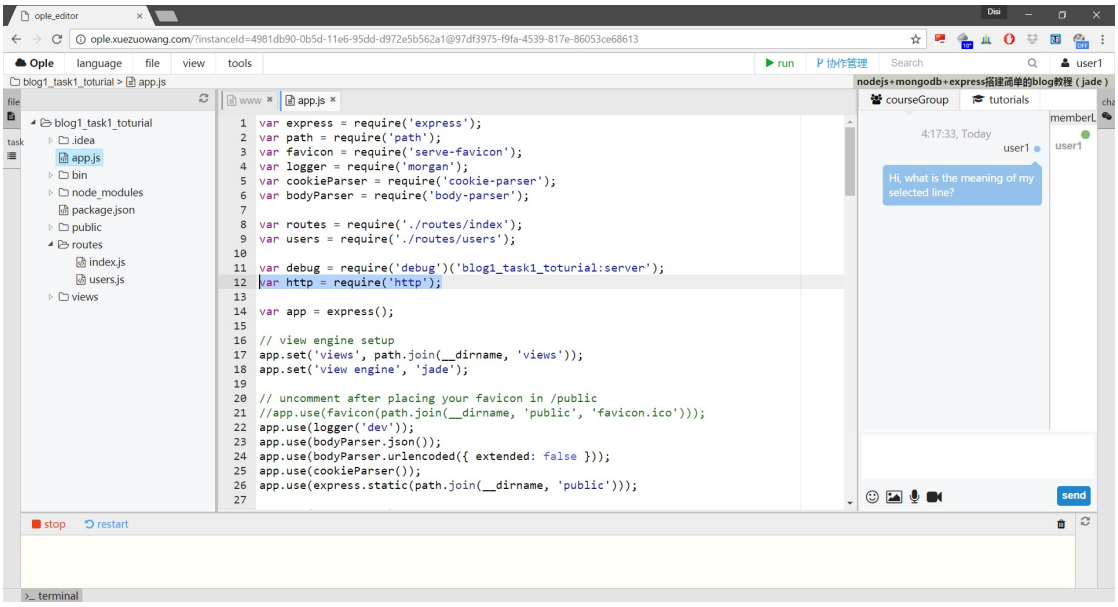
This thesis systematically elaborates an innovative approach to design and implement an integrated design and execution environment for PBL, which will be used to facilitate PBL researchers and practitioners effectively utilizing state of art ICTs to achieve their PBL pedagogical goals.

The future work can have two branches. One branch is to keep the focus on the technology enhanced PBL research. From this point of view, we can conduct more evaluations that are not only for the authoring tool but also for the native runtime tool and the engine since there are still some important features not yet evaluated and the PBL scripting language still needs to be improved. Moreover, as the results of the conducted evaluations suggest, there is a lot of improvement work that needs be done. Additionally, some other important new functions can also be considered. For example, (1) we can research the way to make the script transformation steps automatic. In other words, the system may be able to extract learning processes directly from existing semi-structured textural scripts, and then store and represent them through the diagram-based process model or even directly run the textural scripts; (2) we can research and develop a new search engine that recognizes similar processes according to the given scripts in order to help PBL designers to find other people's process models or retrieve models more efficiently. We can call this new search function a pattern based search.

The other branch is to apply the same model-driven approach presented in this thesis to other systems. In fact, this approach is applied to design and develop a workflow-based learning system in a Chinese company. The goal of this system is to help people learn programming language on a Web-based integrated Web application development environment. In this system, a workflow authoring tool



(a) The workflow authoring tool for defining the knowledge structure and the learning path.



(b) The web-based programming tool as the learning environment.

Figure 10.1: The two most important tools of a workflow-based learning system makes use of the same model-driven approach presented in this thesis.

(Figure 10.1a) and a Web-based programming tool (Figure 10.1b) is needed. The authoring tool defines the knowledge structure and the learning path for a programming language learning tutorial; while the Web-based programming tool as an online learning environment controlled by the learning path provides learners a handy practice place for learning. The feedback of the system development and the early usage test results show that the model-driven and the engineering methodology presented in this thesis is also suitable for this kind of use case. However, formal evaluation is also needed, which may be the other point of our feature work.

Bibliography

Adelsberger, H. H., Collis, B., and Pawlowski, J. M. (2013). *Handbook on information technologies for education and training*. Springer Science & Business Media.

ADLI et al. (2001). Sharable content object reference model (scorm^{TM}). *Advanced Distributed Learning*.

Ali, Z. and Samaka, M. (2013). epbl: Design and implementation of a problem-based learning environment. In *Global Engineering Education Conference (EDUCON), 2013 IEEE*, pages 1209–1216. IEEE.

Ali, Z., Wang, D., Samaka, M., and Miao, Y. (2016). Plate-pbl: Development and implementation of a script-based pbl environment in moodle. In *16th IEEE International Conference on Advancing Learning Technologies*. IEEE.

Aßmann, U., Zschaler, S., and Wagner, G. (2006). Ontologies, meta-models, and the model-driven paradigm. In *Ontologies for software engineering and software technology*, pages 249–273. Springer.

Atkinson, C. and Kuhne, T. (2003). Model-driven development: a metamodeling foundation. *IEEE software*, 20(5):36–41.

Barron, B. and Darling-Hammond, L. (2008). Teaching for meaningful learning: A review of research on inquiry-based and cooperative learning. book excerpt. *George Lucas Educational Foundation*.

Barrows, H. and Kelson, A. (1995). Problem-based learning in secondary education and the problem-based learning institute. *Springfield, IL: Problem-Based Learning Institute*, 1(1):1–5.

Barrows, H. S. (1986). A taxonomy of problem-based learning methods. *Medical education*, 20(6):481–486.

Barrows, H. S. (1996). Problem-based learning in medicine and beyond: A brief overview. *New directions for teaching and learning*, 1996(68):3–12.

Barrows, H. S., Tamblyn, R. M., et al. (1980). *Problem-based learning: An approach to medical education*. Springer Publishing Company.

Beetham, H. and Sharpe, R. (2013). *Rethinking pedagogy for a digital age: Designing for 21st century learning*. routledge.

Bennett, S., Lockyer, L., and Agostinho, S. (2004). Investigating how learning designs can be used as a framework to incorporate learning objects. In *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, pages 116–122.

Bennett, S. J., Agostinho, S., and Lockyer, L. (2005). Reusable learning designs in university education.

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., and Rumble, M. (2012). Defining twenty-first century skills. In *Assessment and teaching of 21st century skills*, pages 17–66. Springer.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5):61–72.

Boud, D. (1985). *Problem-based learning in education for the professions*. Higher Education Research and Development Society of Australasia.

Boud, D. and Feletti, G. (1997a). *The challenge of problem-based learning*. Psychology Press.

Boud, D. and Feletti, G. (1997b). Changing problem-based learning: Introduction to the second edition. *The challenge of problem-based learning*, pages 1–14.

Bower, G. H., Hilgard, E. R., et al. (1981). *Theories of learning*. Prentice-Hall.

Bransford, J., Goldman, S., Pellegrino, J., et al. (1991). Some thoughts about constructivism and instructional design. *Educational Technology*, 31(9):16–18.

Bransford, J. D., Brown, A. L., and Cocking, R. R. (1999). *How people learn: Brain, mind, experience, and school*. National Academy Press.

Bransford, J. D., McCarrell, N. S., Franks, J. J., and Nitsch, K. E. (1977). Toward unexplaining memory. *Perceiving, acting, and knowing: Toward an ecological psychology*, pages 431–466.

Britain, S. (2004). A review of learning design: concept, specifications and tools. *A report for the JISC E-learning Pedagogy Programme*, 2006.

Buneman, P. (1997). Semistructured data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 117–121. ACM.

Center, P. P. R. (2010). 21st century skills for students and teacher.

Conole, G. and Oliver, M. (2006). *Contemporary perspectives in e-learning research: themes, methods and impact on practice*. Routledge.

Conole, G., Oliver, M., Falconer, I., Littlejohn, A., Harvey, J., and Conole, G. (2007). Designing for learning. *Contemporary perspectives in e-learning research: themes, methods and impact on practice', part of the Open and Distance Learning Series, F. Lockwood,(ed), RoutledgeFalmer*.

Cunningham, D. (1992). Assessing constructions and constructing assessments: A dialogue in duffy. *TM & Jonassen, DH Construtivism and the Technology of Instruction-A Conversation. LEA Publishers*.

Dalziel, J. et al. (2003). Implementing learning design: The learning activity management system (lams).

Darling-Hammond, L., Barron, B., Pearson, P. D., Schoenfeld, A. H., Stage, E. K., Zimmerman, T. D., Cervetti, G. N., and Tilson, J. L. (2015). *Powerful learning: What we know about teaching for understanding*. John Wiley & Sons.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340.

Dede, C. (2010). Comparing frameworks for 21st century skills. *21st century skills: Rethinking how students learn*, 20:51–76.

Derntl, M., Neumann, S., and Oberhuemer, P. (2011). Community support for authoring, sharing, and reusing instructional models: the open graphical learning modeler (openglm). In *Proceedings of IEEE ICALT*, volume 2011, pages 431–435.

Derry, S. and Hmelo-Silver, C. (2005). Reconceptualizing teacher education: Supporting case-based instructional problem solving on the world wide web. *Technology-based education: Bringing researchers and practitioners together*, pages 21–38.

Derry, S. J., Hmelo-Silver, C. E., Nagarajan, A., Chernobilsky, E., and Beitzel, B. D. (2006). Cognitive transfer revisited: Can we exploit new media to solve old problems on a large scale? *Journal of Educational Computing Research*, 35(2):145–162.

Dillenbourg, P. (2002). Over-scripting cscl: The risks of blending collaborative learning with instructional design. *Three worlds of CSCL. Can we support CSCL?*, pages 61–91.

Duffy, T. M. and Jonassen, D. H. (1992). Constructivism: New implications for instructional technology. *Constructivism and the technology of instruction: A conversation*, pages 1–16.

edrawsoft (2016). Flowchart benefits. https://www.edrawsoft.com/flowchart-benefits.php. Accessed: 2016-12-07.

Ellis, R. K. (2009). Field guide to learning management systems. *ASTD Learning Circuits*, page 2009.

EML (2002). Educational modelling language 1.1 (eml 1.1). http://dspace.ou.nl/handle/1820/80. Accessed: 2016-11-28.

Ertmer, P. A. and Newby, T. J. (2013). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly*, 26(2):43–71.

Ertmer, P. A. and Simons, K. D. (2006). Jumping the pbl implementation hurdle: Supporting the efforts of k–12 teachers. *Interdisciplinary Journal of Problem-based learning*, 1(1):5.

Europ. (2014). *Terminology of European education and training policy: A selection of 130 key terms*. Office for Official Publ. of the Europ. Communities.

Evensen, D. H. and Hmelo-Silver, C. E. (2000). *Problem-based learning: A research perspective on learning interactions*. Routledge.

Gallagher, S. A., Stepien, W. J., and Rosenthal, H. (1992). The effects of problem-based learning on problem solving. *Gifted Child Quarterly*, 36(4):195–200.

Gijselaers, W. (1995). Perspectives on problem-based learning. In *Educational Innovation in Economics and Business Administration*, pages 39–52. Springer.

Griffin, P., McGaw, B., and Care, E. (2012). *Assessment and teaching of 21st century skills*. Springer.

Griffiths, D., Beauvoir, P., Barrett Baxendale, M., Hazlewood, P., and Oddie, A. (2007). Development and evaluation of the reload learning design editor.

Griffiths, D., Beauvoir, P., Liber, O., and Barrett-Baxendale, M. (2009). From reload to recourse: learning from ims learning design implementations. *Distance Education*, 30(2):201–222.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928.

Gulfcoast (2016). Problem based learning. https://www.gulfcoast.edu/current-students/academic-divisions/social-sciences/epi/pbl/. Accessed: 2016-11-22.

Harrer, A., Malzahn, N., and Hoppe, H. U. (2007). Graphical modeling and simulation of learning designs. *Frontiers in Artificial Intelligence and Applications*, 162:291.

Hernández-Leo, D., Asensio-Pérez, J. I., Derntl, M., Prieto, L. P., and Chacón, J. (2014). Ilde: community environment for conceptualizing, authoring and deploying learning activities. In *European Conference on Technology Enhanced Learning*, pages 490–493. Springer.

Hernández-Leo, D., Villasclaras-Fernández, E. D., Asensio-Pérez, J. I., Dimitriadis, Y., Jorrín-Abellán, I. M., Ruiz-Requies, I., and Rubia-Avi, B. (2006). Collage: A collaborative learning design editor based on patterns. *JOURNAL OF EDUCATIONAL TECHNOLOGYAND SOCIETY*, 9(1):58.

Hmelo, C. E. (1998). Problem-based learning: Effects on the early acquisition of cognitive skill in medicine. *The journal of the learning sciences*, 7(2):173–208.

Hmelo, C. E., Holton, D. L., and Kolodner, J. L. (2000). Designing to learn about complex systems. *The Journal of the Learning Sciences*, 9(3):247–298.

Hmelo-Silver, C. E. (2004). Problem-based learning: What and how do students learn? *Educational psychology review*, 16(3):235–266.

Hmelo-Silver, C. E., Derry, S. J., Bitterman, A., and Hatrak, N. (2009). Targeting

transfer in a stellar pbl course for pre-service teachers. *Interdisciplinary Journal of Problem-based Learning*, 3(2):4.

Hmelo-Silver, C. E., Duncan, R. G., and Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: A response to kirschner, sweller, and clark (2006). *Educational psychologist*, 42(2):99–107.

Hmelo-Silver, C. E. and Eberbach, C. (2012). Learning theories and problem-based learning. In *Problem-based learning in clinical education*, pages 3–17. Springer.

IMS (2003). Ims learning design specification. https://www.imsglobal.org/learningdesign/index.html. Accessed: 2016-11-22.

IMSA (2016). Problem-based learning design institute. https://www.imsa.edu/extensionprograms/problem-based-learning. Accessed: 2016-11-22.

JISC (2006). Recourse ld editor. http://www.reload.ac.uk/new/ldeditor.html. Accessed: 2016-11-30.

Johnson, P. A. (1999). Problem-based, cooperative learning in the engineering classroom. *Journal of Professional Issues in Engineering Education and Practice*, 125(1):8–11.

Jonassen, D., Davidson, M., Collins, M., Campbell, J., and Haag, B. B. (1995). Constructivism and computer-mediated communication in distance education. *American journal of distance education*, 9(2):7–26.

Jones, B. F. et al. (1994). Designing learning and technology for educational reform.

Kaldoudi, E., Bamidis, P., Papaioakeim, M., and Vargemezis, V. (2008). Problem-based learning via web 2.0 technologies. In *Computer-Based Medical Systems, 2008. CBMS'08. 21st IEEE International Symposium on*, pages 391–396. IEEE.

Karagiorgi, Y. and Symeou, L. (2005). Translating constructivism into instructional design: Potential and limitations. *Educational Technology & Society*, 8(1):17–27.

Karampiperis, P. and Sampson, D. (2005). Designing learning services for open learning systems utilizing ims learning design. In *Proceedings of the 4 th IASTED International Conference on Web-Based Education (WBE 2005)*, pages 21–23.

Kay, K. (2010). 21st century skills: Why they matter, what they are, and how we get there. *21st century skills: Rethinking how students learn*, pages 2091–2109.

Kent, S. (2002). Model driven engineering. In *International Conference on Integrated Formal Methods*, pages 286–298. Springer.

Kirschner, P. A., Sweller, J., and Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist*, 41(2):75–86.

Kleppe, A. G., Warmer, J. B., and Bast, W. (2003). *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional.

Koehler, M. J. and Mishra, P. (2005). Teachers learning technology by design. *Journal of computing in teacher education*, 21(3):94–102.

Koper, R. (2001). Modeling units of study from a pedagogical perspective: the pedagogical meta-model behind eml.

Koper, R. and Manderveld, J. (2004). Educational modelling language: modelling reusable, interoperable, rich and personalised units of learning. *British Journal of Educational Technology*, 35(5):537–551.

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

Koper, R. and Tattersall, C. (2005). Learning design. a handbook on modelling and delivering networked education and training.

Laanpere, M., Pata, K., Normak, P., and Põldoja, H. (2012). Pedagogy-driven design of digital learning ecosystems: the case study of dippler. In *International Conference on Web-Based Learning*, pages 307–317. Springer.

Levy, F. and Murnane, R. (2007). How computerized work and globalization shape human skill demands. *Learning in the global era: International perspectives on globalization and education*, pages 158–174.

Liddle, S. W. (2011). Model-driven software development. In *Handbook of Conceptual Modeling*, pages 17–54. Springer.

Lingnau, A., Kuhn, M., Harrer, A., Hofmann, D., Fendrich, M., and Hoppe, H. U. (2003). Enriching traditional classroom scenarios by seamless integration of interactive media. In *Advanced Learning Technologies, 2003. Proceedings. The 3rd IEEE International Conference on*, pages 135–139. IEEE.

Linn, M. C., Clark, D., and Slotta, J. D. (2003). Wise design for knowledge integration. *Science education*, 87(4):517–538.

Linn, M. C., Shear, L., Bell, P., and Slotta, J. D. (1999). Organizing principles for science education partnerships: Case studies of students' learning about 'rats in space'and 'deformed frogs'. *Educational Technology Research and Development*, 47(2):61–84.

Lockyer, L., Bennett, S., Agostinho, S., and Harper, B. (2009). Handbook of research on learning design and learning objects: issues, applications, and technologies (2 volumes). *IGI Global, Hershey, PA*.

Martel, C., Vignollet, L., Ferraris, C., David, J.-P., and Lejeune, A. (2006). Modeling collaborative learning activities on e-learning platforms. In *ICALT*, pages 707–709. Citeseer.

Maurer, H. and Neuhold, C. (2012). Problems everywhere? strengths and challenges of a problem-based learning approach in european studies. In *Strengths and Challenges of a Problem-Based Learning Approach in European Studies. APSA 2012 Teaching & Learning Conference Paper*.

Mayo, P., Donnelly, M. B., Nash, P. P., and Schwartz, R. W. (1993). Student perceptions of tutor effectiveness in a problem-based surgery clerkship. *Teaching and Learning in Medicine: An International Journal*, 5(4):227–233.

McAndrew, P., Nadolski, R., and Little, A. (2005). Developing an approach for learning design players. *Journal of Interactive Media in Education*, 2005(1).

McLoughlin, M. and Darvill, A. (2007). Peeling back the layers of learning: A classroom model for problem-based learning. *Nurse Education Today*, 27(4):271–277.

Mellor, S. J., Clark, T., and Futagami, T. (2003). Model-driven development: guest editors' introduction. *IEEE software*, 20(5):14–18.

Miao, Y. (2000). *Design and implementation of a collaborative virtual problem-based learning environment*. PhD thesis, TU Darmstadt.

Miao, Y. (2005). Cosmos: Facilitating learning designers to author units of learning using ims ld. In *ICCE*, pages 275–282.

Miao, Y., Hoeksema, K., Hoppe, H. U., and Harrer, A. (2005). Cscl scripts: Modelling features and potential use. In *Proceedings of th 2005 conference on*

Computer support for collaborative learning: learning 2005: the next 10 years!, pages 423–432. International Society of the Learning Sciences.

Miao, Y. and Koper, R. (2007). An efficient and flexible technical approach to develop and deliver online peer assessment. In *Proceedings of the 8th iternational conference on Computer supported collaborative learning*, pages 506–515. International Society of the Learning Sciences.

Miao, Y., Samaka, M., and Wang, D. (2014). Plate workbench: A pbl authoring tool. In *11rd European ConFerence on Technology Enhanced Learning*. IEEE.

Miao, Y., Samaka, M., and Wang, D. (2015a). A problem-oriented approach to represent and manage knowledge in pbl. In *23rd International Conference on Computers in Education*. IEEE.

Miao, Y., Samaka, M., Wang, D., Ali, Z., and Romanowski, M. (2015b). Using a pbl authoring tool to train teachers in designing an online pbl unit. In *Proc. of the 22nd International Conference on Computers in Education–ICCE 2014*.

Miao, Y., Wang, D., Nongho, M. F.-P., and Samaka, M. (2015c). Towards a web-based adaptive problem-based learning application. In *15th IEEE International Conference on Advanced Learning Technologies*. IEEE.

Mills, D. (2006). Problem-based learning. *The Higher Education Academy. Retrieved*, 15.

Mills, J. E., Treagust, D. F., et al. (2003). Engineering education—is problem-based or project-based learning the answer. *Australasian journal of engineering education*, 3(2):2–16.

Needham, D. R. and Begg, I. M. (1991). Problem-oriented training promotes spontaneous analogical transfer: Memory-oriented training promotes memory for training. *Memory & cognition*, 19(6):543–557.

Neumann, S. and Oberhuemer, P. (2009). User evaluation of a graphical modeling tool for ims learning design. In *International Conference on Web-Based Learning*, pages 287–296. Springer.

Newby, T., Stepich, D., Lehman, J., and Russell, J. (2000). Instructional technology for teaching and learning: Designing instruction, integrating computers, and using media. *Educational Technology & Society*, 3(2).

Newmann, F. M., Marks, H. M., and Gamoran, A. (1996). Authentic pedagogy and student performance. *American Journal of Education*, pages 280–312.

Noy, N. F., McGuinness, D. L., et al. (2001). Ontology development 101: A guide to creating your first ontology.

Olivier, B. (2006). Learning design update.

OMG (2003). Mda guide version 1.0.1.

OMG (2014). Mda guide version 2.0.

OUNL (2006). Copperauthor. https://sourceforge.net/projects/copperauthor/?source=navbar. Accessed: 2016-11-30.

OUNL (2008). Coppercore. http://coppercore.sourceforge.net/. Accessed: 2016-11-30.

P21 (2007). P21 framework for 21st century learning. http://www.p21.org/storage/documents/docs/P21_Framework_Definitions_New_Logo_2015.pdf. Accessed: 2016-11-14.

Paquette, G., de la Teja, I., Léonard, M., Lundgren-Cayrol, K., and Marino, O. (2005). Using an instructional engineering method and a modeling tool. *Learning Design: Modelling and Implementing Network-based Education & Training, Ed (s) R. Koper & C. Tattersall, C. Springer-Verlag.*

Paquette, G., Léonard, M., Lundgren-Cayrol, K., Mihaila, S., Gareau, D., et al. (2006). Learning design based on graphical knowledge-modelling. *Educational Technology & Society*, 9(1):97–112.

Pellegrino, J. W., Hilton, M. L., et al. (2013). *Education for life and work: Developing transferable knowledge and skills in the 21st century.* National Academies Press.

Peter F. Oliva, W. R. G. (2013). *Developing the Curriculum, 8th Edition.* Peachpit Press.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach.* Palgrave Macmillan.

Prieto, L. P., Asensio-Perez, J. I., Munoz-Cristóbal, J. A., Dimitriadis, Y. A., Jorrín-Abellán, I. M., and Gomez-Sanchez, E. (2013). Enabling teachers to

deploy cscl designs across distributed learning environments. *IEEE Transactions on Learning Technologies*, 6(4):324–336.

Purichia, H. (2014). Problem-based learning: An inquiry approach. *Interdisciplinary Journal of Problem-Based Learning*, 9(1):1.

Putnam, R. T. and Borko, H. (2000). What do new views of knowledge and thinking have to say about research on teacher learning? *Educational researcher*, 29(1):4–15.

QTI, I. (2005). Ims question & test interoperability specification. *IMS Global Learning Consortium*.

Ryberg, T., Glud, L. N., Buus, L., and Georgsen, M. (2010). Identifying differences in understandings of pbl, theory and interactional interdependencies. In *Networked Learning Conference 2010*, pages 943–951.

Rychen, D. S. and Salganik, L. H. (2003). *Key competencies for a successful life and well-functioning society*. Hogrefe Publishing.

Rychen, D. S. E. and Salganik, L. H. E. (2001). *Defining and selecting key competencies.* hogrefe & huber publishers.

Samaka, M., Miao, Y., and Wang, D. (2016). Support peer assessment processes in online problem-based learning. In *Global Engineering Education Conference (EDUCON), 2016 IEEE*, pages 488–497. IEEE.

Sardo-Brown, D. (1990). Experienced teachers' planning practices: A us survey. *Journal of education for teaching*, 16(1):57–71.

Savery, J. R. (2015). Overview of problem-based learning: Definitions and distinctions. *Essential readings in problem-based learning: Exploring and extending the legacy of Howard S. Barrows*, pages 5–15.

Savery, J. R. and Duffy, T. M. (1995). Problem based learning: An instructional model and its constructivist framework. *Educational technology*, 35(5):31–38.

Schank, R. C. and Abelson, R. P. (2013). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Schmidt, D. C. (2006). Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2):25.

Schmidt, H. G. and Moust, J. H. (2000). Factors affecting small-group tutorial

learning: A review of research. *Problem-based learning: A research perspective on learning interactions*, pages 19–52.

Schwartz, D. L. and Bransford, J. D. (1998). A time for telling. *Cognition and instruction*, 16(4):475–5223.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE software*, 20(5):19.

Seymour, A. (2010). Managing group dynamics and developing team working in problem-based learning. *Problem-based learning in health and social care*, pages 67–78.

Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., DeWitt, D. J., and Naughton, J. F. (1999). Relational databases for querying xml documents: Limitations and opportunities. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 302–314. Morgan Kaufmann Publishers Inc.

Sharples, M., Adams, A., Alozie, N., Ferguson, R., FitzGerald, E., Gaved, M., McAndrew, P., Means, B., Remold, J., Rienties, B., et al. (2015). Innovating pedagogy 2015: Open university innovation report 4.

Sharples, M., Taylor, J., and Vavoula, G. (2005). Towards a theory of mobile learning. In *Proceedings of mLearn*, volume 1, pages 1–9.

Shaw, M. and Garlan, D. (1996). *Software architecture: perspectives on an emerging discipline*, volume 1. Prentice Hall Englewood Cliffs.

Spiro, R. J., Feltovich, P. J., Jacobson, M. J., and Coulson, R. L. (1991). Knowledge representation, content specification, and the development of skill in situation-specific knowledge assembly: Some constructivist issues as they relate to cognitive flexibility theory and hypertext. *Educational technology*, 31(9):22–25.

Stahl, T., Voelter, M., and Czarnecki, K. (2006). *Model-driven software development: technology, engineering, management*. John Wiley & Sons.

Stepien, W. and Gallagher, S. (1993). Problem-based learning: As authentic as it gets. *Educational leadership*, 50:25–25.

Szabo, M. and Flesher, K. (2002). Cmi theory and practice: Historical roots of learning management systems.

Tabbers, H., Kester, L., Hummel, H., and Nadolski, R. (2004). Interface design for digital courses. *Integrated E-learning: Implications for pedagogy, technology and organization*, pages 100–111.

TENCompetence (2010). Recourse. https://tencompetence-project.bolton.ac.uk/ldauthor/. Accessed: 2016-11-30.

Trilling, B. and Fadel, C. (2009). *21st century skills: Learning for life in our times*. John Wiley & Sons.

Van Deursen, A., Klint, P., and Visser, J. (2000). Domain-specific languages: An annotated bibliography. *Sigplan Notices*, 35(6):26–36.

Villasclaras-Fernández, E. D., Hernández-Gonzalo, J. A., Hernández-Leo, D., Asensio-Pérez, J. I., Dimitriadis, Y., and Martínez-Monés, A. (2009). Instancecollage: A tool for the particularization of collaborative ims-ld scripts. *Journal of Educational Technology & Society*, 12(4):56.

Voelter, M., Benz, S., Dietrich, C., Engelmann, B., Helander, M., Kats, L. C., Visser, E., and Wachsmuth, G. (2013). *DSL engineering: Designing, implementing and using domain-specific languages*. dslbook. org.

Wang, D., Miao, Y., Hoppe, U., and Samaka, M. (2014a). A domain-specific modeling language approach to support various forms of online pbl. In *14th IEEE International Conference on Advanced Learning*. IEEE.

Wang, D., Miao, Y., Hoppe, U., and Samaka, M. (2014b). A flexible system and data model for the representation and management of pbl scripts. In *22nd International Conference on Computers in Education*. IEEE.

Wang, D., Samaka, M., Miao, Y., Ali, Z., and Hoppe, H. U. (2016). A model-driven pbl application to support the authoring, delivery, and execution of pbl processes. *Research and Practice in Technology Enhanced Learning*, 11(6):1.

Woods, D. R. (1996). Problem-based learning for large classes in chemical engineering. *New Directions for Teaching and Learning*, 1996(68):91–99.

Woods, D. R. (2000). Helping your students gain the most from pbl.

Woods, D. R. (2006). *Preparing for PBL, 3rd edition*. McMaster University.

Yadav, A., Subedi, D., Lundeberg, M. A., and Bunting, C. F. (2011). Problem-based learning: Influence on students' learning in an electrical engineering course. *Journal of Engineering Education*, 100(2):253.

Yinger, R. (1979). Routines in teacher planning. *Theory into practice*, 18(3):163–169.

Appendix A

The Scenarios for the Evaluation

A.1 Biological scenario: Deformed Frogs

- Twenty learners are involved in a seminar.
- The learners are divided into three groups.
- The lesson consists of four phases.
- Each learner has a networked computer.

Hints If you cannot find the same term and vocabulary in the authoring tool, you can choose a similar term and vocabulary.

Phase 1

Type Problem engagement

Goal Support learners as they become engaged in the problem with personal enthusiasm and interest.

Steps 1) The teacher presents to all learners in the class about the discovery of deformed frogs near a local area lake. 2) All learners view some articles with pictures and a video titled "The Frogs – What Are They Really Telling Us?" on a website (for example: http://www.deformed_frog.com). (Note: Such problem sources challenge learners to investigate the status of

the frog population; it also encourages them to take a proactive stand on this environmental concern.)

Phase 2

Type Problem definition, Aiming and planning

Goal Support learners to identify and decompose learning goals according to the KNK charts and to create effective information-gathering and information-sharing plan.

Steps 1) Teacher couches all learners to define a problem. 2) All learners prioritize the needs to know (according to importance) and identify the prerequisite relation among them. (Note: structure learning needs) 3) They set and decompose learning goals. 4) They define a set of learning actions to achieve the goals. 5) They allocate tasks to groups.

Phase 3

Type Research

Goal Support learners in gathering data, acquiring knowledge, and understanding how new information contributes to an understanding of the problem and how information is assessed in light of its contribution to that understanding.

Steps 1) According to the action plan, three groups work in parallel. a) Group 1: Learners in this group individually gather information using mobile devices when they observe soil and water tests, look for industry pollution sources, count frog population, etc., in order to help them understand the environmental factors affecting frogs. b) Learners in this group collaboratively analyse the investigation results and write an analysis report. a) Group 2: Learners in this group collaboratively develop a questionnaire for interviews using a brainstorming tool. b) Then they individually interview persons such as the frog experts and others concerned. c) They return to their group room to analyse the results of the interviews and to write an analysis report together. a) Group 3: Learners in this group individually gather information from articles, books, videos, web sites, and other resources. 2) All learners integrate information in a panel session action.

Phase 4

Type Problem resolution

Goal Support learners in articulating problems and issues under the circumstances they are given and in identifying conflicts. Support learners to develop solutions.

Steps 1) Learners individually generate tentative solutions. 2) All learners discuss the problem situation and their solutions. 3) Because they find that they have different opinions regarding solutions, they use what they learned to debate different perspectives.

A.2 Physical scenario: Getting Attention at an Intersection

Phase 1: Problem engagement

On the first day of the unit the teacher presents the students with the problem statement:

> You are a member of a team investigating a problem of drivers speeding through red lights at intersections. You are responsible for gathering scientific data and creating a solution to this problem via presentation or a model in which your team examines the consequences of speeding through a red light and how the current red light system leads to this traffic problem.

Then the students watch a video showing an intersection on a highway. This video contains the highway lanes, intersection, red lights at the intersection, and traffic and shows three vehicles going through the intersection with a red light in the oncoming lanes.

Phase 2: Problem definition, identification of knowledge gaps & planning

As a class the students make lists of facts (what we know), ideas (possible reasons for cars going through the lights at the wrong time), learning issues (what we need to know more about), and actions (what we need to do), recording their thoughts on large sheets of poster paper tacked to the classroom wall. The teacher facilitates the large-group brainstorming sessions to keep students focused and to guide them to resources that will answer their questions.

Phase 3: Research

After phase 2, the students break off into 4 small groups of 4-5 students and begin independent online research about current red light systems used at intersections. Groups will research current legal codes for red lights and current warning systems at intersections. After taking some time to research red lights, students will research and learn about kinematics of vehicles traveling through the intersections. During this phase, topics pertaining to kinematics and electrical systems will be taught by the teacher with some time given to the schedule to allow for the group work.

Phase 4: Experiment & Evaluate

Based on their research results groups perform the lab called "Series and Parallel Circuits" in which they construct and analyze different types of electrical circuits. A group hypothesis is formulated on how to make an intersection safer. Groups will then determine and apply a method to collect data to test their hypothesis. They evaluate the collected data to support or refute their hypothesis.

Phase 5: Report

The student groups create multimedia presentation or build models to be presented to the panel of judges which will consist of community members, local politicians, parents, and a highway department official.

A.3 Medical scenario: The Heart and Circulatory System

Dr. Winter teaches a course on "the heart and circulatory system" at the university hospital Charité in Berlin. To familiarize his students with cardiac insufficiency he prepared four different descriptions of cases with patients suffering from cardiac insufficiency. There are four extent levels of cardiac insufficiency (NYHA I to NYHA IV) and each of the cases addresses one of these levels. The case descriptions only contain the patients' reasons for visiting a doctor and an initial anamnesis.

Dr. Winter arranges his twenty students into four groups, each consisting of five students. Each group is provided with one case description (PDF).

After reading the description on their own the students should engage in a group discussion using the video conference system "Adobe Connect" that is provided by the university and can be accessed via web browser. Each student group has an own video conference room. The goal of the group discussion is to identify the relevant facts from the case description (e.g. symptoms and risk factors), generate first ideas regarding the causes of the symptoms and identify information needed for stating a diagnosis. Furthermore, they should create an action plan how to fill their knowledge gaps and distribute the required tasks among the group members. All results (facts, ideas, information needs, action plan) are captured on the white board that is provided by the video conference software.

Information needs can arise from knowledge deficiencies regarding disease patterns that match with the symptoms, but also from the incomplete case descriptions. Thus, on the one hand students are provided with links to medical guidelines as well as an eBook on "the heart and circulatory system". On the other hand, they can ask Dr. Winter via mail for specific examination and diagnostic test results (e.g. blood tests) regarding their cases.

In the next lecture, they should present their case, the identified facts and their reasoning results to the complete course. To collaboratively prepare slides the groups use Google Drive. They agree on a plan for creating the slides and also recorded this plan on the white board. Additionally to creating slides, after the group meeting each student writes a case summary as well as a summary of relevant medical knowledge for solving this case. This summary is submitted to Dr. Winter. After the presentation of the slides to the course, there is a discussion with all

students, in which Dr. Winter exposes the true diagnosis. Furthermore, he records the relevant facts for this diagnosis on the electronic white board.

Appendix B

Questionnaires in the Evaluation

B.1 Error Tolerance

- If I make a mistake, I can easily restore everything to its previous state.
- My impression is that very little effort is involved in correcting mistakes.
- The lesson consists of four phases. The software provides me with useful information on how to recover from error situations.
- I perceive the error messages as helpful.

B.2 Suitability for the Task

- The functions implemented in the software support me in performing my work.
- The software forces me to perform tasks that are not related to my actual work. (transformed)
- I perceive the arrangement of the fields on-screen as sensible for the work I do with the software.
- Too many different steps need to be performed to deal with a given task. (transformed)
- In a given screen, I find all of the information I need in that situation.

- The important commands required to perform my work are easy to find.
- The presentation of the information on the screen supports me in performing my work.

B.3 Self-Descriptiveness

- I understand immediately what is meant by the messages displayed by the software.
- It is easy to retrieve information about a certain function.
- The software provides not only general explanations but also concrete examples to illustrate points.
- The software displays basic information about conceptual aspects of the program.
- The software provides me with enough information about which actions are permitted in a particular situation.
- I can tell straight away which functions are invoked by specific commands.
- The terms and concepts used in the software are clear and unambiguous.
- I can tell straight away which functions I have to use to achieve an intended goal.
- In the usage of the system, I encountered various situations where I didn't know what to do next. (transformed)

B.4 Controllability

- The possibilities for navigation within the software are adequate.
- The software makes it easy for me to switch between different menu levels.
- I can interrupt any dialog at any time.
- Facilities of the software support optimal usage of the system functionality.

B.5 Conformity with User Expectations

- I can anticipate which screen will appear next in a processing sequence.
- Terms and graphical elements are used consistently in all parts of the software.
- I find that the same function keys are used throughout the program for the same functions.
- When executing functions, I have the feeling that the results are predictable.

B.6 Learnability

- I needed a long time to learn how to use the software. (transformed)
- The explanations provided by the system help me understand the software so that I become more and more skilled using it.
- I was able to use the software right from the beginning by myself, without having to ask for help.
- I feel encouraged by the software to try out new system functions by trial and error.
- In order to use the software properly, I must remember many details.
- I find it easy to recall the effect of functions and commands provided by the software.

Appendix C

Content of the Attached CD

A CD is attached with this thesis with the following content:

- Source code of the IDEE4P system.
- The PDF version of this thesis.
- More documents of the evaluation.

DOI: 10.17185/duepublico/47830
URN: urn:nbn:de:hbz:464-20190506-094205-3