A novel Approach for Designing an E-Learning Pattern Language

Maysoon Aldhekhail, Mohammed Alawairdhi, Alaa Eldeen Ahmed, and Azeddine Chikh

***Abstract*—** Recently, e-learning has become a very important topic, and researches are focusing on improving it using both the technology and the pedagogical domain theories. Design patterns in e-learning are descriptions of good practice in e-learning. It focuses on providing a solution to a learning problem in such a way that designers can use this solution a million times over without ever doing it the way twice. Experts from different disciplines are supposed to use these patterns for different objectives related to their community. These objectives may need to use several patterns together and consequently form a pattern language, which can help in solving a group of interrelated problems. In this paper we introduce a new mechanism for building a learning design pattern language for designers who are using IMS-LD as standard specification. All the existing works consider the designer as a main actor and so they use the bottom-up approach in building the language. We add experts as another actor and so we use a top-down approach to build a pattern language and a bottom-up approach to build patterns. Results show that our approach gets more stable results.

***Keywords*—**learning patterns, Pattern languages, E-learning, XML, IMSLD, learning workflow

I. INTRODUCTION

ELEARNING is a growing to be more common in educational organizations. Learning design focuses on activities done by participants instead of content. Towards getting an efficient increase in the quality and variety of e-learning, three central features should be considered [1]:

The first is the learning activity, where people always learn better when they are actively involved in doing something. These activities could be in the form of discussions, simulations, problem-solving exercises, role-plays, quizzes or meta-learning tasks such as mind-maps. The second is The creating learning workflow, which is achieved by giving thoughts to the sequential order and timing of the various activities and the presentation of the resources needed to support them. The thirds is the sharing and re-using learning design. In other words, the learning design needs to be described at a sufficient level of abstraction so that it can be generalized beyond the single learning context for which it is created.

Both theoretical and practical knowledge about e-learning is monotonically increasing so that educational institutions can gain from this knowledge in developing and running e-learning courses.

Since e-learning is based on sharing contents and resources, there are many international standards for sharing educational design and integrating digital courses. In this paper we use the IMS Learning Design (IMS-LD) specification, which is a standardized computer language developed specifically for describing educational processes and has many advantages compared to other learning design specifications[2][3][4]. These advantages can be briefed as Completeness, Pedagogical Flexibility, compatibility, reusability, formalization and Reproducibility [5], [0]. IMS-LD helps teachers and designers organize and orchestrate their learning activities efficiently, which can then be reused by other teachers and designers.

The concept behind the design patterns and the pattern languages can help in extracting and solve repeated instructional design problems in e-learning [6]. Design patterns are a very effective way to capture proven solutions for recurrent problems. It is mainly used to externalize the implicit knowledge of an expert, using a highly structured description format. The notion of a pattern language is proposed in [6]. It is defined as a collection of related patterns that captures the whole of the design process and can guide the designer through step-by-step design guidelines. In this paper we build a pattern language that merges both the designer and expert knowledge to improve the language performance level.

This paper is organized as follows. In section 2, patterns are described. In section 3 we present the definition of pattern languages. In section 4 learning patterns are introduced. The pattern language is proposed in section 4. Finally, the conclusion recalls the main contribution and opens some future perspectives.

II. IMS LEARNING DESIGN (IMS-LD) SPECIFICATION

In this paper we focus on We focus on the IMS Learning Design (IMS-LD) specification, which is a standardized computer language developed specifically for describing educational processes.

A. Objectives of IMS-LD specification

In terms of objectives, the IMS-LD specification aims to

- Provide a containment framework of elements that can describe any design of a teaching-learning process in a formal way [4].
- Integrate the activities of both the learners and teachers.
- Integrate the resources and services used during learning.
- Support a wide variety of pedagogical approaches such as problem-based learning, competence-based learning and game-based learning.
- Supports mixed mode (i.e. blended learning) as well as pure online learning.
- Enable authors to specify the complete learning design of a course with all its details explicitly, instead of selecting a restricted set of hardwired designs like in the LMS. This means that the designer can specify the type, sequence and way of learning activities, also the desired interaction between different persons in different roles.
- Captures processes rather than content.

B. Requirements of IMS_LD specification

The IMS-LD specification is developed to meet some specific requirements [5], [0]:

- Completeness: The specification must be able to fully describe the teaching-learning process in a unit of learning, including references to the digital and non-digital learning objects and services needed during the process.
- Pedagogical Flexibility: It must be able to describe different kinds of pedagogies without prescribing any specific pedagogical approach.
- Personalization: It must be able to describe personalization aspects within a learning design, so that the content and activities within a unit of learning can be adapted based on the preferences, pre-knowledge, educational needs and situational circumstances of users. In addition, it must allow the designer, when desired, to pass the control over the adaptation process to the learner, a staff member and/or the computer.
- Compatibility: It must enable learning designs to use and effectively integrate other available standards and specifications where possible.
- Reusability: It must make it possible to identify, isolate, and exchange useful learning objects, and to re-use them in other contexts.
- Formalization: It must describe a learning design in the context of a unit of learning in a formal way, so that automatic processing is possible.
- Reproducibility: It must enable a learning design to be abstracted in such a way that repeated execution, in different settings and with different persons.

C. Components of IMS-LD specification

IMS-LD consists of several components [5]:

- Conceptual Model: For the description of a teaching-learning process and defines the basic concepts and relations in IMS-LD, it is expressed as Unified Modeling Language UML model.
- Information Model: To specify exactly how the entities in conceptual model relate to each other, it is the core document of the specification [5].
- Best Practice and Information Guide: Specifies some use cases and best practices.
- Binding: Technology used to implement information model is a series of XML schema.

III. PATTERN, PATTERN LANGUAGE AND LEARNING PATTERN

E-learning patterns are focused to produce mechanisms to help in the design of learning materials and systems, considering the same principles initially established for architectural design patterns. There are three important three main concepts that need to be understood in order to produce a good design. These concepts are pattern, pattern language and learning pattern:

A. Pattern

A pattern is an abstract solution to a problem in a certain context. The primary goal of patterns is to create an inventory of solutions to help in resolving problems that are common, difficult and frequently encountered. The idea of pattern was originally introduced by architect Christopher Alexander and his colleagues at the end of the 70s as a way to describe solutions to reoccurring problems encountered in architectural design. The goal was to support both architects and general public in designing quality towns, neighborhoods and homes [7]. Now, design pattern is a new idea in the field of human computer interaction and educational technology to support designers in interaction and instructional design [8].

Using Pattern has many benefits such as allowing exchange of expertise with others, giving a chance to novice users to learn from experts, presenting strategies regarding common recurring decisions, support reusability, provide more flexible solutions than static templates and finally, save time for designers. Many formats for design patterns are available. All these formats share some common minimum characteristics such as pattern name, problem description, context and the solution itself. Some of these design patterns are: Alexandrian pattern [5], E-LEN pattern model [9] and GOF Pattern model [10], [11],[9]. To develop any of the previous patterns, there is a development life cycle as shown in Fig. 1 [12]:

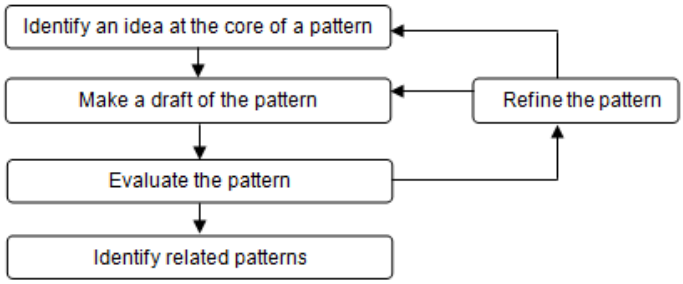

Figure1. The pattern development life cycle

As shown in the figure, the first step is to identify the idea of the core pattern (pattern mining). There are two classifications of methods for pattern mining [13]. The first is called Inductive pattern mining, which is the most used method. It is based on the derivation of general principles from particular facts or examples related to the problem. The second is called Deductive pattern mining that is heavily based on thinking (mental) processes, observation and experts experience [13].

Once we have defined a draft for the pattern, we need to evaluate the pattern. This may be achieved by collecting suggestions. Collecting suggestion may be done either by following the work of Neil Harrison the work of members of the E-LEN team [12]. In Neil Harrison method, the author should describe several issues such as pattern ownership, matching degree between the problem and the solution. The other way is focused on validating several checklist items to evaluate pattern [12]. Checklist items may include questions such as "Does the pattern contain a recognizable problem, which occurs over and over again in your professional practice?" or "Is the name of the pattern meaningful? Can you guess what the pattern might be about based only on the pattern name?".

Once patterns are captured, related patterns are identified by Checking whether the pattern contains more than one solution, if so, it should be more than one pattern. Also, a check is needed to find out if there is a lower level patterns that are needed to complete or elaborate on an existing pattern.

B. Pattern Language

Patterns need language to describe it. Salingaros defined a pattern language as: “A pattern is an encapsulation of forces; a general solution to a problem. The language combines the nodes [patterns] together into an organizational framework” [5]. This definition highlights the underlying hierarchical nature of a pattern language. It considers that they are the connectivity rules between patterns that make a collection of patterns into a language.

The first developed pattern language was “A pattern Language: Towns, Buildings, Construction” was published in 1977 by the architect Christopher Alexander et al. He introduced 253 patterns in the architectural domain presenting patterns for everything from designing independent regions to cities, to buildings and even to designing single rooms. By connecting these patterns with common forces and other relations, he transformed this collection of pattern to a pattern language [5].

Pattern languages are needed to provide guidance on how to successfully use combinations of patterns from a collection. Also, they are needed to provide a way of understanding, and possibly controlling, a complex system. Finally, they provide the order in which problems should be solved.

C. Learning Pattern

Previous sections discussed the concept of pattern and pattern language, and the different application domains of them. In this paper, we will clarify and focus on the learning design domain to build an e-learning pattern language. Because people are the central focus of learning, learning patterns have to deal with biological and social basics that cannot be ignored. Several aspects of good learning that must be considered when identifying and evaluating learning pattern, such as Learning is active, Learning is individual, Learning is cumulative, Learning is self-regulated, Learning is goal-oriented, Learning is situated and Learning can be learned[14]. There are different classifications of patterns in instructional design process such as pedagogical, learning experiences, activity, interaction and content [[14]].

IV. THE PROPOSED LANGUAGE PATTERN

All the existing works for building a pattern language is based on the bottom-up approach (from patterns to pattern language). This approach considers the designer as the main actor. In our proposed approach, we look at the pattern language as a joint point between the designers and experts. In other words, pattern language is considered as a shared space between designers and experts. In general some designers face some difficulties or needs and experts will look up for designers needs. Some other designers propagate their best practices and recommendations and Experts can use them as a reference to build new patterns. As a consequence, we consider the pattern language as a space where needs are expressed, best practices are proposed and patterns (as solutions) are offered. For these reasons, we use a top-down approach to build a pattern language and a bottom-up approach to build patterns. We can consider a pattern as composed of three main components:

Pattern = Problem + Solution + Metadata

The solution is the main component and considered the most important. Metadata contains other information and divided into two parts: Constraint that is mandatory and Optional which contains extra useful information. The proposed work aims to offer learning designers who use IMS-LD specification, double support: both academic and practical.

The first one proposes ontology (LDO: learning Design Ontology) which will provide theoretical knowledge about concepts related to learning design. The second one proposes a pattern language (LDPL: learning Design Pattern Language) which will provide experiential knowledge about different learning design aspects.

LDO and LDPL systems will be structured at the higher level according to IMS-LD structure. Indeed the elements (Activity, Role, Environment and Method) will be the main concepts which will structure the remainder knowledge.

LDPL support which is the part concerned by this research follow these specifications:

1- Approach: Top-down and then Bottom-up. We defined the architecture first then we added pattern knowledge as components.
2- Representation :XML standard
3- Life cycle : Prototyping model
4- Development method : Object oriented method with UML notation
5- XML Schema
6- Prototyping

Now we describe the conceptual model of learning design of the pattern language (LDPL) system attributes for each class, the logical model of the pattern language, the idea of the pattern language, then the xml schema.

A. Conceptual Model of Learning Design Pattern Language (LDPL)

The following figure shows the conceptual model of Learning Design Language (LDPL) of the proposed system. The designer explains his problems or difficulties and the expert figures what the designers are really facing and helps them. The designer also can benefit from other designers by considering the more common problems and takes advantages of their comments. According to figure 2, the designer searches for a solution for his problem and views similar problems. If no solution is proposed, the problem is added to the pattern language. The expert now will know the needs of designers and try to figure solutions for their problem. Moreover, other designers can recommend some comments that may lighten the problems.

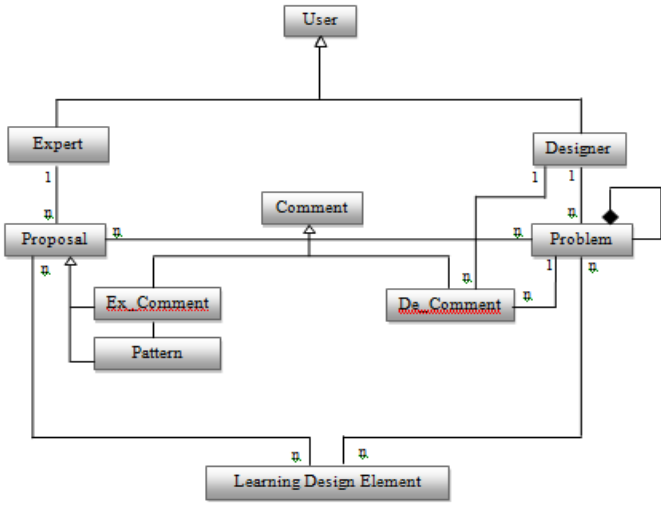

Figure 2. Conceptual model of learning design pattern language

As in the Fig. 2, Designer get into LDPL system to finding proposed solution of their problems and viewing similar problems. If the proposed solution for the problem was not found, it will be added to the pattern language. The expert now will know the needs of designers and try to figure out solutions for the problem .Moreover, other designers can recommend some comments that may lighten the problems.

Experts offer their experience to designers and see what they need and what difficulties facing them. They propose solutions to designer's problems using a powerful strategy (patterns) with formal structure .this will explain in more details how solution will be used and problems that may face when trying to apply these patterns. Experts can create a new pattern for one or more problems, transfer existing solution into formal pattern or link between existing patterns and problems. They also advice some comments which help the designer to solve their problems.

Designer can present the problems with any type: pedagogical, technical, IMS_LD specifications or any other problems.

As in Fig. 2, conceptual model contains many classes which describe different parts of the pattern language. Each class has a various attributes show their characteristics and functions. To give the designer more reliability of proposed solutions in pattern language, we provided detailed information about the experts.

The pattern class includes two parts mandatory and optional. The mandatory part may be represented by any of the pattern models described in the previous section. To give the pattern more flexibility, we defined an optional part contain any other attributes the expert wants. patterns can be classified into different kinds and we can use these classification to categorize problems and consequently its proposals in the pattern language. These categories have three classes (pedagogical, technical and IMS_LD specification). The Pedagogical class contains academic or education problems and proposals. The Technical class means the technical problems that may face the designer, e.g. problem in using authoring tool of IMS_LD specification. IMS_LD specification class which contains problems related to using IMS_LD specification and their elements. To provide flexibility to the pattern language "other" category is added. The category will be used when the problem or proposal does not belong to any of the main three categories.

Table1 shows different classes in LDPL system and their attributes with some comments to explain ambiguous attribute.

Class	**Attributes**	**Comments**
User	First Name	
	Last Name	
	E-mail address	
	Company or University	
	Username	

	Password	
Expert	Expert ID	
	Highest qualification held	
	Total Experience	
	Functional Area	
	Current Industry	
	Key skills	
	C.V	Attached as file
Designer	Designer ID	
Problem	Problem ID	
	Problem Title	
	Problem Description	
	Problem Date	
	Problem Type	Technical, Pedagogical, IMS-LD Specification or other.
	Teaching Problem Domain	Science, Mathematics, History…,etc
Proposal	Proposal Description	
	Proposal Date	
	Proposal Type	Technical, Pedagogical, IMS-LD Specification or other.
Comment	Comment Author	
	Comment Content	
Expert Comment	Expert Comment ID	
Designer Comment	Designer Comment ID	
Pattern	Pattern ID	Unique identifier for the pattern
	Name	Name for the pattern.
	Problem	Description of a problem
	Solution	The solution itself
	Context	Explains when to apply the pattern
	Forces	Describe the trade-offs of applying the solution.
	Related Pattern	All related patterns put here.
	Author(s)	The authors of the pattern.
	References	All references
	Other elements	Optional

Table 1: Attributes of LDPL classes

B. Logical Model (XML binding of LDPL)

The work in this paper is based on exchanging data between designers and experts using the web so, the most suitable choice to represent this work is XML. We employ an XML schema to describe the structure of LDPL, the relations between classes, elements, and attributes for each class. Here, we present an example of XML schema that gives a complete image of expert's role in the pattern language.

The following code segment in table 2 shows the Expert definition in XML schema. Expert is 'subclass' of user class therefore, it inherences the attributes of user class (First Name, Last Name, E-mail address, Company or University, Username and Password). Expert's curriculum vitae (C.V) was defined as a type of any URI which means that expert can specify his C.V as URI and has two roles in pattern language, Advice_Comment which defines the relation between the expert and his comment and Propose_Pattern which defines the relation between expert and his proposed pattern.

```
<!-- ..... Expert
..... ->
<xs:complexType name="Expert">
<xs:complexContent>
<xs:extension base="User">
<xs:sequence>
   <xs:element name="Highest-Qulification-held"
type="xs:string"/>
   <xs:element name="Total-Experience" type="xs:string"/>
   <xs:element name="Functional-Area" type="xs:string"/>
   <xs:element name="Current-Industry" type="xs:string"/>
   <xs:element name="Key-Skill" type="xs:string"/>
   <xs:element name="C.V" type="xs:anyURI"/>
   <xs:element name="advice_Comment"
type="advice_relation" minOccurs="0"
maxOccurs="unbounded"/>
  <xs:element name="propose_Pattern"
type="propose_relation" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Expert_ID" type="xs:ID"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Table 2: Experts part of XML schema

Since pattern is the core of pattern language; the pattern part of XML schema is represented as follows:

```
<!--
.....Pattern.....
..... -->
<xs:complexType name="Pattern">
<xs:complexContent>
<xs:extension base="Proposal">
   <xs:sequence>
```

```
   <xs:element name="Pattern-name" type="xs:string"/>
    <xs:element name="Pattern-Problem" type="xs:string"/>
    <xs:element name="Solution" type="xs:string"/>
    <xs:element name="Context" type="xs:string"/>
   <xs:element name="Forces" type="xs:string"/>
   <xs:element name="Related-Pattern" type="xs:string"/>
   <xs:element name="Author" type="xs:string"/>
   <xs:element name="References" type="xs:string"/>
   <xs:element name="SolveProblem"
type="solvedBy_relation" minOccurs="0"
maxOccurs="unbounded"/>
   <xs:element name="For_LD_element" type="for_relation"
minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="Proposed_byExpert"
type="propose_relation"/>
   <xs:any minOccurs="0"/>
    </xs:sequence>
<xs:attribute name="Pattern_ID" type="xs:ID"/>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

Table 3 Pattern part of XML schema

Refereeing to the Conceptual model, Pattern is a subclass of proposal therefore; it's defined as an extension. Pattern ID is defined as a unique and the elements of the pattern are defined as sequence elements. Pattern contains three types of relations, "SolveProblem", "For_LD_element" and "Proposed_ByExpert". "SolveProblem" defines the relation between the pattern and the problem it is solving. Pattern can solve one or more of designer's problems or it may not relate to any existing designers problems. "For_LD_element" describes the relation between the pattern and the LD element if pattern solved a problem related to LD element. "Proposed_ByExpert" defines the expert who adds the pattern to LDPL system. Different elements could be added by expert, when needed. Pattern may relate to one or many problems. Problem is related to pattern through solved_by_Pattern relation.

The xml schema of the problem can be defined as follows:

```
<!-- .....
Problems..... -->
<xs:complexType name="Problem">
<xs:sequence>
   <xs:element name="Problem-Title" type="xs:string"/>
   <xs:element name="Problem-Description"
type="xs:string"/>
   <xs:element name="Proplem-Date" type="xs:date"/>
   <xs:element name="Problem-Type" type="Problem-
Category"/>
   <xs:element name="Teaching-Problem-Domain"
type="xs:string"/>
   <xs:element name="Course-Audience" type="xs:string"/>
   <xs:element name="has_Designer" type="has_relation"/>
   <xs:element name="facilitateBy_De_Comment"
type="facilitateBy_relation" minOccurs="0"
maxOccurs="unbounded"/>
   <xs:element name="In_LD_element" type="In_relation"
minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="solved_by_pattern"
type="solvedBy_relation" minOccurs="0"
maxOccurs="unbounded"/>
   <xs:element name="lighted_by_Ex_comment"
type="lighted_relation" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Problem_ID" type="xs:ID"/>
</xs:complexType>
```

Table 4: Problem's definition in XML schema

C. XML Instance of XML Schema

To illustrate the pattern language (LDPL), we build an XML file from XML schema. As an example of XML Instance, we take the same explained parts in XML schema (expert, pattern and problem). Table 5 shows these parts:

```
<Expert Expert_ID="ID_1">
   <First-Name>Dr. Azeddine </First-Name>
   <Last-Name>CHIKH</Last-Name>
   <Email-Address>az_chikh@KSU.EDU.SA</Email-
Address>
   <Company-OR-Uiversty>King Saud University
</Company-OR-Uiversty>
   <Password>CHIKH1965</Password>
   <Highest-Qulification-held>PHD  in Computer
Science</Highest-Qulification-held>
   <Total-Experience>21years</Total-Experience>
   <Functional-Area>Researching,Teaching</Functional-
Area>
   <Current-Industry> Associate Professor, Information
Systems Department , College of computer engineering, King
Saud University</Current-Industry>
   <Key-Skill>writing</Key-Skill>
   <C.V>http://faculty.ksu.edu.sa/chikh/Pages/cvEnglish.aspx
</C.V>
   <advice_Comment advice_ID="ID_88"/>
   <propose_Pattern propose_ID="ID_20"/>
   <propose_Pattern propose_ID="ID_30"/>
</Expert>
-----------------------------------------------------------------
-----------------------------------------------------------------
---------
<Pattern Pattern_ID="ID_111">
      <Proposal-Description>This pattern will solve the
problem for answering student's questions</Proposal-
Description>
      <Proposal-Date>2004-05-12</Proposal-Date>
       <Proposal-type>
   <Pedagogical/>
       </Proposal-type>
       <Pattern-name>FAQ</Pattern-name>
        <Pattern-Problem>Students have problems and
questions that necessitate quick responses.</Pattern-
Problem>
       <Solution>
```

```
Create a document in the course that contains a list of questions along with the answers.This include questions that have been already asked by students, and if you have taught the course before, you can include common questions from previous students.  It would also be beneficial to include any questions that you anticipate students may have. To insure that students utilize this document, encourage them to refer to it as a resource tool for them to address many of their concerns.
    </Solution>
    <Context>
Any web-based course consisting of students whose location may be different from that of the instructor's or who are novice web students.
    </Context>
    <Forces>
          Emails from students can quickly fill up an instructor's email account.
          Student's work hours may be different from instructor's hours.
          Students want quick responses
    </Forces>
    <Related-Pattern>Feedback-Loop</Related-Pattern>
    <Author>Jon Smith</Author>
    <References>
          Khan, B. (Ed). (1997). Web-Based Instruction
          Shaw, R. (1996). The FAQ Manual of Style.
    </References>
   <solveProblem solvedBy_ID="ID_444"/>
   <Proposed_byExper propose_ID="ID_20"/>
-------------------------------------------------------------------------------------------------------------
<Problem Problem_ID="ID_11">
   <Problem-Title>Answering students questions</Problem-Title>
   <Problem-Description>Students have problems and questions that necessitate quick responses</Problem-Description>
   <Proplem-Date>2003-12-23</Proplem-Date>
   <Problem-Type>
      <Pedagogical/>
    </Problem-Type>
   <Teaching-Problem-Domain>Any</Teaching-Problem-Domain>
   <Course-Audience>Any</Course-Audience>
   <has_Designer has_ID="ID_100"/>
   <solved_by_pattern solvedBy_ID="ID_444"/>
   <lighted_by_Ex_comment Lighted_ID="ID_555"/>
 </Problem>
```

Table 5: Example of XML instance

In this example, expert advised one comment and two patterns. Assume the pattern is called (FAQ pattern). FAQ pattern used to solve the problem of quick necessary responses of student problems and difficulties. The solution proposed by the pattern author is to create a document in the course that contains a list of questions and answers. This includes questions that have been already asked by students, and if teacher have taught the course before, he could include common questions from previous students. It would also be beneficial to include any predicted questions from students. This pattern is related to problem through the relation SolveProblem and connects to the expert who adds it to the pattern language through Proposed_byExpert relation.

V. LDPL Tool Kit

In order to validate and test the proposed LDPL, we implemented a tool kit with the following specification.

A. Problem:

How can the designer and the expert share learning resources in order to build effective learning patterns instead of being limited to reading separate resources in individual ways.

B. Solution:

Provide a toolkit for both the designer and the expert to use, which will facilitate them in studying and exploring the existing similar problems and their solutions. Then they can decide which solution that is closer to their own preferences. This tool should offer them a set of functions for the user (expert or designer).

1) Designer Part functionalities

The designer can benefit from LDPL system in different ways. He can use the system to search for solution of specific problem she faced or to add a comment to problem.

Also she can view all problems in the system and their proposed solutions if any. The toolkit has two main menu items:

a) Problem: designer can add new problem or list existing problems.

b) Proposal: designer can view all proposals in the system.

The first one is related to the problem the need to be solved. Through these items the designer is able to:

- View problem's information with its solutions (if any).
- Access these solutions (patterns, expert comments).
- View similar specific problem.
- Update the problem, if he is legal (only problem's author is legalized to update the problem) designer can edit and then click Ok.
- Delete a problem, if he is legal, by clicking Delete button.
- View designer's comment for this problem.
- Add new comment to the problem (by entering comment's title and description)

Also, Designers can enter the problem metadata (all information except problem description) to the system. The tool kit allow designer to explore any similar problems, their patterns and their comments to help the user to enhance his design. If there is no previous solutions proposed for the problem, the designer will be allowed to add the problem description so that other may be able to use it later.

2) Expert Part functionalities

Expert can benefit from the toolkit in different aspects. He can use the LDPL system to see what the designer's problems are and what the learning design difficulties that designers face. In addition he can present the proposals either as pattern

or comment in the system. Expert interface has three functionalities:

a) Problem: where expert can view all problems in the system either solved or not.
b) Proposal: where expert can add new proposal or view existing ones; proposal can be pattern or expert comment.
c) Tool: where he can: View and edit expert's account, Export a pattern from the pattern language, and Import a pattern to the pattern language.

Also experts are able to display pattern information, update a pattern or even delete a pattern.

VI. Conclusion

In this paper presented a new approach for building pattern language in e-learning. As far as we know, all existing work present the pattern language as a collection of patterns proposed by expert and related to each other in a specific way. When designers search for solutions of their designing problems it takes great effort. In other hand, experts present solutions as patterns without knowing what the actual problems designers face are. LDPL is considered as an interaction environment, between designers who use IMS-LD specification, and experts.

Reference

[1] Sue Bennett, Shirley Agostinho, Lori Lockyer, Lisa Kosta,Jennifer Jones,Rob Koper,Barry Harper "Learning designs: Bridging the gap between theory and practice", Acsilite, pp.51-60, 2007.
[2] Colin Tattersall, Tim Sodhi, Daniel Burgos, Rob Koper "Using the IMS Learning Design notation for the modelling and delivery of education". Handbook of Visual Languages for Instructional Design: Theories and Practices, pp. 299-314, Oct 2006.
[3] Rob Koper, Yongwu Miao. "Using the IMS LD Standard to Describe Learning Design", 2006.
[4] Chew, L. K. "IMS Learning Design and eLearning" Second International Conference on eLearning for Knowledge-Based Society. Bangkok, Thailand, 2005
[5] Koper, R "Current Research in Learning Design" Educational Technology & Society, pp. 13-22, 2006.
[6] Alexander, C., Ishikawa, S., & Silverstein, M "A pattern language: towns, buildings, construction" New York: Oxford University Press, 1977.
[7] Paris Avgeriou,Andreas Papasalouros,Symeon Retalis,Manolis Skordalakis "Towards a Pattern Language for Learning Management Systems" Educational Technology & Society, pp.11-24, 2003.
[8] Sherri S Frizell, R. H "Aligning Theory and Web-based Instructional Design Practice with Design Pattern", Proceedings of E-Learn-World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education , pp. 298–304, 2003.
[9] Tutorail Making e-learning design patterns. http://www2.tisip.no/E-LEN/tutorial/index.html
[10] GoF's Design Patterns. http://channah.com/portfolio/talk/Paper/GoF-patterns/GoFTemplate.htm
[11] A.Salingaros, N. "The Structure of Pattern Language. arq-Architectural Research Quarterly" pp. 149-161, 2000.
[12] E.-L. p.team. "Design patterns and how to produce them".E_LEN,2004.
[13] Baggetun, R., Rusman, E. & Poggi, C. "Design Patterns For Collaborative Learning: From PracticeTo Theory And Back" In L. Cantoni & C. McLoughlin (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications pp. 2493-2498, 2004.
[14] Manuel Caeiro, Martín Llamas, Luis Anido. "E-learning Patterns : an Approach to Facilitate the design of E-learning Materials", CIIEE, pp. 294-303, 2007.

Maysoon Aldhekhail obtained her B.Sc. degree in computer applications from King Saud University, Saudi Arabia in 2001. Then she received his M.Sc. degree in information Systems from King Saud University, Saudi Arabia in 2009. Her master dissertation was in e-learning patterns and systems.
She works as lecturer in the college of Computer and Information Sciences, Al - Imam Muhammad ibn Saud Islamic University. She is also the vice-chair of the information systems department.
Her research interests include e-learning (patterns, environments), e-commerce, and HCI (usability, adaptability).

Mohammed A. Alawairdhi obtained his B.Sc. degree in information Systems from King Saud University, Saudi Arabia in 1996. Then he received his M.Sc. degree in Computer Science from California State University, Chico, USA in 2000. He earned his Ph.D. degree in Computer Science in 2009 from De Montfort University, Leicester, UK. His Ph.D. research was in software engineering.
He works as an Assistant Professor in the college of Computer and Information Sciences, Al - Imam Muhammad ibn Saud Islamic University. He is also the vice-dean of Graduate Studies and Research and head of the college research center.
His research interests include software evolution, ubiquitous computing (sensor-based applications, sensor networks, smart environments, and RFID), business process reengineering (modeling, simulation), and Human Computer Interaction (culture and cyberspace).

Alaa Eldeen S. Ahmed obtained his B.Sc. degree in Communication Engineering from Zagazig University, Egypt in 1993. Then he received his M.Sc. degree in Computer Science in Cairo University, Cairo, in 1998. He earned his Ph.D degree in computer Science in 2005 from University of Connecticut, USA. His Ph.D was in fault tolerant Real time scheduling.
He works as Assistant Professor in the college of computer and information sciences, Al – Imam Muhammad ibn Saud Islamic University.
His research interests include Distributed systems support for harvesting unused resources in cloud computing environment, fault tolerant algorithm for mobile agents, wireless sensor network performance evaluation and the management of E-learning.

Azeddine Chikh obtained his B.Sc. degree in in Computer Science National Institute of Computer Science, Algeria in 1988. He earned his MSc. In Computer Science National Institute of Computer Science, Algeria in collaboration with University of Claude Bernard at Lyon – France in 1994. He earned his Ph.D degree in Computer Science from the National Institute of computer science, Algeria in collaboration with University of Paul Sabatier, Toulouse, France in 2004.
He works as an Associate Professor in Information Systems Department, college of computer and information sciences, King Saud University. He worked as an Associate Professor, Faculty of engineering, University of Aboubakr Belkaid, Tlemcen, Algeria.
His research interests include Decision Sciences, e-learning, Information System, Ontology, Pattern Recognition, Semantic Web Technologies, Software Engineering.