A Top down Approach for Designing an E-learning Pattern Language based on IMS-LD Specification

MAYSOON ALDEKHAIL, ALAA ELDEEN SAYED AHMED, MOHAMMED ALAWAIRDHI, AZEDDINE CHIKH

College of Computer and Information Sciences
Al-Imam Muhammad Ibn Saud Islamic University, King Saud University
SAUDI ARABIA

Maysoon, alaaeldeen.sayed, awairdhi@ccis.imamu.edu.sa, az_chikh@ksu.edu.sa

Abstract

Designing an efficient e-learning system is a multipart problem that includes several issues such as designing learning tasks, learning resources and social forums that will enable each student to learn in a good and effective way. Learning Design patterns are focused on producing mechanisms to design the learning material in such a way that is neither too prescriptive nor based on a single model. Applying good learning design patterns on the E-learning paradigm can help both designers and experts to narrow the bridge between theory and practice of learning design. In this paper we use a top down approach for designing an e-Learning design pattern to help designers, who are using IMS Learning Design (IMS-LD) standard specification, in sharing and exchanging their best practices in learning design.

Keywords: learning patterns, Pattern languages, E-learning, XML, IMSLD, learning workflow

1. Introduction

E-learning is 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 [6]:

1. 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 metalearning tasks such as mind-maps.
2. 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.
3. 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][4][12]. These advantages can be briefed as Completeness, Pedagogical Flexibility, compatibility, reusability, formalization and Reproducibility [10], [7].

The concept of the design patterns and the pattern languages can help to extract and solve repeated instructional e-learning design problems [1].

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.

2. Patterns’ definitions

The most powerful definition of pattern was coined by the architect C. Alexander as it conveys all central aspects of a pattern: “Pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution of that problem, in such a way that you can use this solution a million times over, without ever repeating it in the same way twice” [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 [9], E-LEN pattern model [15] and GOF Pattern model [14], [9],[15]. To develop any of the previous patterns, there is a development life cycle as shown in (figure 1) [7]:

![Fig1. The pattern development life cycle](image)

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 [4].
1. Inductive pattern mining, which is the mostly used method. It is based on the derivation of general principles from particular facts or examples related to the problem.
2. Deductive pattern mining that is heavily based on thinking (mental) processes, observation and experts experience [4].

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 [7]. 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 [7]. 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.

3. Pattern languages
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” [9]. 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 [9].

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.

4. Learning Patterns
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[2]. There are different classifications of patterns in instructional design process such as pedagogical, learning experiences, activity, interaction and content [2].

5. The proposed language pattern
From our point of view, a pattern language is not just a collection of patterns which are solutions of some problems. It is more than that; it is a joint point between the 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. All the existing works are built according to the bottom-up approach (from patterns to pattern language). In order to get more stability, we aim to 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:

The solution is the main component of a pattern. The metadata is divided into two parts: mandatory and optional

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.

![Conceptual Model of Learning Design Pattern Language](image)

**Fig 2. Conceptual model of learning design pattern language**

As shown in the 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. As in the figure, the conceptual model contains many classes which describe different parts of the pattern language. Classes have various attributes showing their characteristics and functions. Table 1 shows different classes in LDPL and their attributes with some comments to explain ambiguous attributes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>First Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E-mail address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Company or University</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Username</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>Expert</td>
<td>Expert ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Highest qualification held</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total Experience</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Functional Area</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Current Industry</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Key skills</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C.V</td>
<td>Attached as file</td>
</tr>
<tr>
<td>Designer</td>
<td>Designer ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Problem ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Problem Title</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Problem Description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Problem Date</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Problem Type</td>
<td>Technical, Pedagogical, IMS-LD Specification or other.</td>
</tr>
<tr>
<td></td>
<td>Teaching</td>
<td>Science, Mathematics, History...,etc</td>
</tr>
<tr>
<td></td>
<td>Problem Domain</td>
<td></td>
</tr>
<tr>
<td>Proposal</td>
<td>Proposal Description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proposal Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proposal Type</td>
<td>Technical, Pedagogical, IMS-LD Specification or other.</td>
</tr>
<tr>
<td>Comment</td>
<td>Comment Author</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comment Content</td>
<td></td>
</tr>
<tr>
<td>Expert</td>
<td>Expert Comment ID</td>
<td></td>
</tr>
<tr>
<td>Designer</td>
<td>Designer Comment ID</td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td>Designer Comment ID</td>
<td></td>
</tr>
<tr>
<td>Pattern</td>
<td>Pattern ID</td>
<td>Unique identifier for the pattern</td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>Name for the pattern</td>
</tr>
</tbody>
</table>
|              | Problem                          | Description of a
b. 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 Figure 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 anyURI 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.

Pattern part of XML schema

Pattern is a subclass of proposal therefore; it’s defined as an extension. Pattern ID defined it uniquely 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 which
describes the relation between the pattern and the LD element if pattern solved a problem related to LD element. **Proposed_ByExpert** which defines the expert who adds the pattern to LDPL system. There is an Optional part of pattern model was defined as any element with zero minimum occurs. 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.

```
<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="Problem-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" 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:element>
  <xs:attribute name="Problem_ID" type="xs:ID"/>
</xs:sequence>
</xs:complexType>
```

**Problem’s definition in XML schema**

Now, to illustrate the pattern language (LDPL), we can 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). The following Figure 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>
</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
  </References>
```

```


6. 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.

7. Reference


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. To materialize the pattern language we implemented a prototype which illustrates the main functionality of the pattern language and shows how it works.