# Database Support of Web Course Development with Design Patterns

Timothy K. Shih, Dept. of CS and IE, Tamkang University, Taiwan, ROC
Shi-Kuo Chang, Dept. of Computer Science, Univ. of Pittsburgh, USA
Timothy Arndt, Dept. of Computer and Info. Sci., Cleveland State University, USA

## Abstract

*Current distance learning are mostly based on Web technologies. However, course materials announced as Web documents do not have a normalized structure. It is difficult for students to realize where they are in a Web navigation graph. On the other hand, a text book has a fixed structure, such as the hierarchy of chapters and the index. A text book reader knows how to start searching for information with the common structure of books in his/her mind. If distance learning course materials are organized in one or two patterns, it is easier for an individual student to follow. We investigate this approach, and propose a system for Web course designs with patterns. The system also serves as a front end module of a Web learning environment which provides automatic assessment of student performance.*

**Key words**: Distance Learning, Design Pattern

## 1 Introduction

Object-oriented software is reusable, if it is designed properly. Reusability of software components is the key to the success of the object-oriented paradigm. The technology has been developed for many years. Several object-oriented systems also demonstrated the feasibility of such a methodology. As a result, the design experiences of well-engineered object-oriented software systems are shared. A *design pattern* is a reusable experience, which describes a problem that need to be solved over and over again, and proposes a solution to the problem, as well as provides an analysis of the consequence of applying such a solution. Conclusively, a design pattern summarizes experiences of solving problems. There are many design patterns found in good object-oriented software systems[1]. For instance the Iterator design pattern provides a procedure to access a sequence of objects without exposing their internal representation. The Prototype design pattern allows the creation of a new object by copying its prototype. The Composite design pattern allows the composition and decomposition of a hierarchy of objects. These design patterns, as well as many others discussed in [1], are commonly used in object-oriented designs. The most important advantage of using design patterns is, of course, reusability.

In line with the growing of Internet technologies, Web-based distance learning is widely available. Web-based instruction delivery can rely on home pages. The purpose of designing a home page for Web-based distance learning is to teach students. From our experiences, there are several issues to achieve this purpose. For instance, the design of course material, the construction of on-line tutorial, the implementation of on-line monitoring, and the assessment of student performance are some "problems" that may have good "design patterns". To identify and to catalog these design patterns is thus very interesting and important research.

To analyze design patterns of distance learning, it is important to understand that there are several advantages of this new trend of instruction delivery:

- **Flexibility**: spatial/temporal separation of students and instructor
- **Friendliness**: hyperlink traversal of related information instead of the traditional sequential access
- **Scaleability**: references can be easily attached
- **Diversity**: instruction can be delivered in different media
- **Efficiency**: a large number of students can benefit from an instructor, if suitable automatic tools are provided

These advantages make Web-based distance learning attractive. However, when an individual is traversing a Web course, it is easy for one to get lost in the complicated navigation graph, especially when references to other Web sites are given. Even though it is possible for a Web course delivering system to provide some visual guidance of where the student is reading in the course material, however, when the course structure is complicated, it is very difficult to make a clear representation of the structure graph. On the other hand, from the traditional perspective, every reader understands that a book contains a fixed structure pattern, such as the table of contents, the tree structure of chapters, and the indices. A reader looks at the table of contents (or the index) and jumps to a page for specific information. If Web courses can be designed with a fixed pattern, it is easier for the students to understand where they are in the navigation. There is no need of a visual navigation graph since the structure of the Web course is realized.

Our purpose is to investigate what types of patterns, or the combinations of patterns, are commonly used in Web courses. From the experience, we identify a number of basic structures. We then propose a mechanism and system to allow Web course developers to consider using a fixed pattern in their course design. Students registered to a virtual university adapt to only one or two Web course patterns, which is easier for them to remember.

A Web course pattern includes two parts: the *structure pattern* and the *content pattern*. A structure pattern is a composition of basic Web document structures, such as sequences or trees. A section is a node which contains its content in a Web document structure. A content pattern describes the layout and types of multimedia objects of a section.

A *course unit* contains some sections which are organized according to structure pattern and content pattern. The course unit also incorporates with course materials as its content. The basic element in a Web course to be reused is a course unit. A Web course will be constructed according to the following criteria:

- **Small section size**: each section contains multimedia objects which can be displayed in one page, with a unique URL.
- **Limited number of sections in a course unit**: a course unit, such as a lecture, is a basic object which can be reused. A reusable course unit should contain a limited number of sections, which is usually small.
- **Fixed structure**: the organization of sessions in a Web course should follow a fixed structure in a virtual university, or in a Web site of distance learning.
- **Appropriate guidance**: using a Web Navigation Patrol (to be discussed), each course unit will conduct a guidance while a student is traversing the course unit.

Our virtual university supporting system has a tool which allows an instructor to broadcast on-line messages and Web browsing controls to student workstations. It is necessary to simplify a section to one page so that Web browsing can be managed. Also, a course unit can be reused in many ways. An instructor can use a course unit in different courses. In addition, a course unit is a basic object which is used in an intelligent system, which generates individual tutorial for each student based on the learning behavior of such student. In the next section, we propose a system which allows users to implement Web courses according to the above criteria. In this paper, we give an overview of such a system, with an emphasis on the construction of Web course design patterns. Section 2 is an illustration of our system software architecture. The top part of the system architecture is for the construction of Web course patterns. We then discuss the specification of Web course patterns in section 3. The pattern specification is used in the construction of database definition language (DDL) programs, which is presented in section 4. A short conclusion is given in section 5.

## 2 The System Architecture

The proposed system has four parts. In this paper, we focus on the discussion of the first part: the construction of Web course design patterns. The four parts of our system are:

- **The Web course design pattern tools** A number of small tools are used to help the construction of Web course patterns, which will be discussed.
- **The Web Navigation Patrol**: The patrol is a mobile agent program which run on client sites. The agent collects navigation messages of Web interaction, which are used in the analysis model of student behavior.
- **The performance analysis model**: This analysis model is based on some instruction theory from the educational literature, such as the SP-table mechanism with the computation of student caution indices. The out comes of analysis are displayed to the instructor graphically, and are used in the generation of intelligent tutorial for each student.
- **The progressive Learning State Machine Constructor**: The constructor generates intelligent tutorials from the course material provided by the instructor. The generated Web documents are dynamic state machines. The construction intelligence of such a tutorial generation is based on the results of the performance analysis model.

The system can be divided into two parts (see figure 1). The first is a Web course design system and the second is an instruction delivery system. The two systems share the same Web Course Content Database. A virtual university has a curriculum development committee. One of the duties of the committee is to design a Web course pattern specification, which is specified via the assistant of a pattern design editor (i.e., a visual pattern design interface). The semantics of design patterns are checked against a formal syntax of a pattern specification language (discussed in section 3). Database definition language programs are generated and used in the construction of a Web course pattern database. The design pattern contains both the structure and the content templates. A Web course design editor reads the design pattern, guides an instructor to fill in course contents, and generates Web courses. Web courses are used in the instruction delivery system, which is presented by a commercial Web browser, with a delivery control daemon (the Web Navigation Patrol, or WNP). WNP collects Web browser navigation sequences from all students in a Web Navigation Database, which is used by two sub-systems: the Progressive Learning State Machine and the Performance Analysis Model. The first sub-system generates intelligent tutorials, which contains course units extracted from the original Web

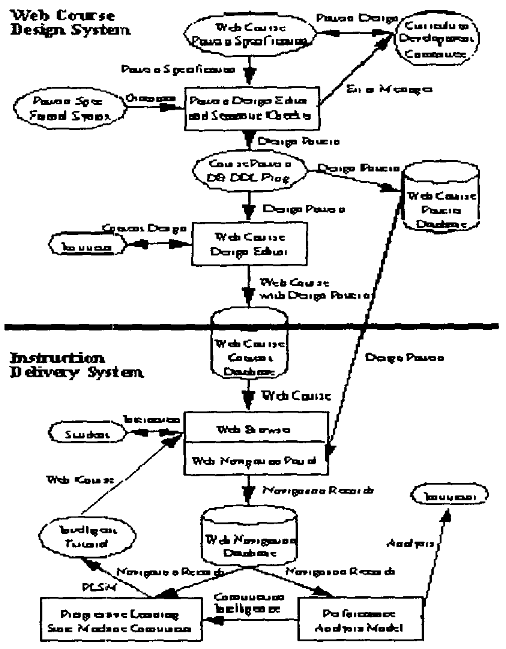

Figure 1: The Virtual University Course Design and Delivery System

course. The second sub-system provide statistic summary for the instructor, so that student learning performance can be evaluated. The details of the instruction delivery system is discussed in another paper. In the next section, we give the definition of a Web course pattern.

## 3 Course Pattern Specification

As discussed in the system architecture (see figure 1), the Web course development committee decide a course pattern for Web courses to be used in a virtual university. The course pattern is specified in a pattern specification, which is designed via a windowing interface. The interface program creates a representation of the pattern according to a formal syntax. The formal syntax only contains part of the pattern information. The system also relies on a Web course design editor to decide the content information, such as a text paragraph or the location of a picture. The formal syntax, on the other hand, does not contain the screen coordinate information. But the syntax gives a formal definition of the structure of a Web course, as well as its content objects.

In general, there are three types of elementary structures in a Web course. Tree structures are commonly used in many Web documents. Tree structures can be used as a text book outline, which can be provided as a class reference to students. List structures can be used in a presentation, which contains some slides. A list structure can be used as an on-line tutorial of some portions of a Web course. A list structure can also represent a lecture sequence. Moreover, class materials may contain some after class references. These references may be Web documents not designed by the instructor of the virtual university. Also, the references may not follow the design pattern. We use sets, which do not have a specific order of their elements, to represent these references.

Using the above three types of basic structures, and an atomic element of the design pattern (i.e., the Session element), the formal specification of a design pattern can be constructed. The formal syntax of design pattern follows:

```
CoursePattern := StructurePattern
StructurePattern := RCU StructureP
RCU := Null
     | CourseUnitName @
CourseUnitName := ASCII
StructureP := Session
            | TreePattern
            | ListPattern
            | SetPattern
TreePattern := Session #< TreeP >
TreeP := Null
       | StructurePattern TreeP
ListPattern := #[ ListP ]
ListP := Null
       | StructurePattern ListP
SetPattern := #{ SetP }
SetP := Null
      | StructurePattern SetP

Session := SessionName SessionContent
           Pop-upQuizzes SessionQuestions
SessionName := ASCII
SessionContent := WebDocument
Pop-upQuizzes := Null
               | Pop-upQuiz Pop-upQuizzes
Pop-upQuiz := YesNoQuestion Constraints
            | MultipleChoiceQuestion Constraints
            | FillInBlankQuestion Constraints
SessionQuestions := Null
                  | SessionQuestion SessionQuestions
SessionQuestion := YesNoQuestion
                 | MultipleChoiceQuestion
                 | FillInBlankQuestion
Constraints := Trigger
             | not Constraints
             | Constraints and Constraints
             | Constraints or Constraints
             | Constraints xor Constraints
Trigger := TimeTrigger
         | ObjectNumberMetTriger
         | SpecificObjectMetTriger
TimeTrigger := TimeVisited >= Integer
ObjectNumberMetTriger := NoOfObjectsVisited >= Integer
SpecificObjectMetTriger := CurrentMObject = MObject
YesNoQuestion := WebDocument
MultipleChoiceQuestion := WebDocument
FillInBlankQuestion := WebDocument

Note:
  Null is a null object
```

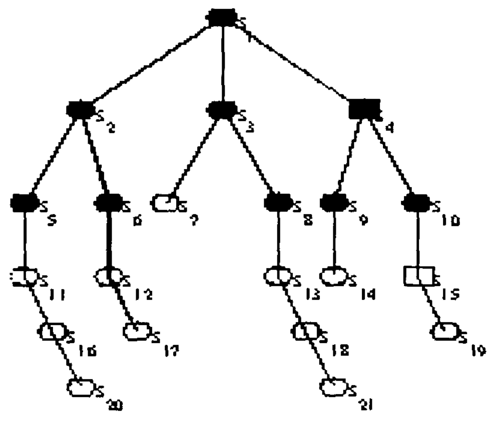

Figure 2: An Example of Structure Pattern

```
  @ represents the structure pattern is
    a reusable course unit
  # is an integer which represents the
    number of internal objects
Special Variables:
  TimeVisited
  CurrentMObject
  NoOfObjectsVisited
Definitions omitted are:
  Integer is an integer
  ASCII is an ASCII string
  MObject is a multimedia object
  WebDocument is an HTML document
```

Each session in a design pattern has a unique name, a session content, and some auxiliary information. Session content are ordinary HTML documents. But, we suggest to the users of the course designer to limit the size of a session to one page (a scrolling page is also suggested). This approach make it easier to design our automatic assessment system, which includes a *Web Navigation Patrol* to collect interaction massages from the Web course users. A session contains two types of tests. A pop-up quiz shows up on a Web browser when some conditions meet. The purpose of these quizzes is to guarantee that a student understand some course materials before they proceed with the next session. A session also has some session questions, which is similar to the home work questions given in a text book. But, session questions can be used in a system which automatically generates exams and tests for a virtual university course. Both pop-up quizzes and session questions are simple style questions with limited answers, such as yes/no questions. However, simple essays are also allowed (but are not suggested for automatic assessment). We have omitted some details in the formal specification, such as ASCII strings and integers.

We use a Web Navigation Patrol in our system to guard student navigation. Three special variables are used. The `TimeVisited` variable is an `Integer`, which denotes the duration of time used by a particular student with respect to a session. The `CurrentMObject` is an `MObject`, which is currently visited. And the `NoOfObjectsVisited` is an `Integer`, which represents the number of `MObject` visited. The three variables keep the navigation status of a student. The instructor is able to use these three variables to design the constraints of pop-up quizzes in a Web course session. The constraints depend on how long the session is visited, how many multimedia objects are visited, and if a particular object is visited. A constraint can contain a logical expression to include the above conditions.

As an example, figure 2 illustrates the structure of a course pattern. We use a solid box or circle to represent object composition. Hollow boxes and circles are atomic objects which are sessions. We use boxes to denote reusable course units, and use circles to represent ordinary sessions. Solid box indicates that, all objects belong to the composition, which is a reuseable course unit. However, a single session may be reused as well. In figure 2, the top part is a tree structure which contains three branches, $S_2$, $S_3$, and $S_4$. each branch has two children. The bottom part of figure 2 contains some linked lists. The composed object, $S_4$, is a reusable course unit, which includes six sessions. However, session $S_{15}$ is a stand alone reusable unit. The formal syntax of the structure pattern follows:

```
Session 3<
  Session 2< 3[ Session Session Session ]
             2[ Session Session ] >
  Session 2< Session
             3[ Session Session Session ] >
  CourseUnitName @
    Session 2< 1[ Session ]
               2[ CourseUnitName @
                    Session Session ] > >
```

Note that, the above example does not contain information about session content patterns. A session has a name and three content components. The following example shows a typical session content pattern:

```
Session :=
   "Introduction to Linked List"
   WebDocument
   MultipleChoiceQuestion TimeVisited >= Integer
                          or CurrentMObject = MObject
   FillInBlankQuestion FillInBlankQuestion
```

The layout of the session is specified in the Web document. One pop-up quiz is used. The quiz will be triggered when the duration of session visiting reaches a certain amount of time, or when a particular multimedia object is visited. In addition, two fill-in-blank questions are included in the session. Note that, the above example is a content pattern. The physical content of the session will be provided by an instructor while he/she is designing a course using such a course pattern.

## 4 Database Specification

Course patterns will be stored in the pattern database. It is possible to automatically generate database definitions from the pattern specification. The structure pattern has three types of atomic structures: tree, list, and set. Sets are represented as lists. And, lists are trees. Therefore, the best strategy to represent a structure pattern is to store every session in a tree, which can be represented as a table. In table 1, **Parent** and **Child** represent a pair of parent-child relation of an edge in a tree. The table also illustrates how the structure pattern specified in figure 2 is stored. The **Category** field of table 1 can be **T** for a tree component, **L** for a list component, and **S** for a set component. The **Reuse** field indicates that an object is reusable. Note that, it is possible to have a tree which is embedded as a node in a list. Similarly, a list may contain a set element. However, a set element contains a number of URLs, with their underlying structure undefined in our structure pattern.

Table 1: The Structure Pattern Table

| Parent | Child | Category | Reuse |
|---|---|---|---|
| S1 | S2 | T | Null |
| S1 | S3 | T | Null |
| S1 | S4 | T | Null |
| S2 | S5 | T | Null |
| S2 | S6 | T | Null |
| S3 | S7 | T | Null |
| S3 | S8 | T | Null |
| S4 | S9 | T | Reuse |
| S4 | S10 | T | Null |
| S5 | S11 | L | Null |
| S11 | S16 | L | Null |
| S16 | S20 | L | Null |
| S20 | Null | L | Null |
| S6 | S12 | L | Null |
| S12 | S17 | L | Null |
| S17 | Null | L | Null |
| S7 | Null | L | Null |
| S8 | S13 | L | Null |
| S13 | S18 | L | Null |
| S18 | S21 | L | Null |
| S21 | Null | L | Null |
| S9 | S14 | L | Null |
| S14 | Null | L | Null |
| S10 | S15 | L | Null |
| S15 | S19 | L | Reuse |
| S19 | Null | L | Null |

Table 2 to table 4 stores content patterns. The primary keys of these tables are the session identifiers. The URLs in these tables are empty in the pattern database. But, the actual URLs are given in the Web course content database. The **Type** field in table 3 and table 4 can be filled with **YN**, **MC**, or **FIB** for different types of questions, as discussed in section 3. Each quiz is associated with a constraint expression stored in the quiz table. A question has three levels of difficulties. The difficulty levels are used in the generation of exams, which will be used in the instruction delivery system.

Table 2: The Content Pattern Table

| Session | Name | Start URL |
|---|---|---|
| S1 | Introduction to Linked List | ... |
| ... | | |

Table 3: The Quiz Table

| Session | Type | URL | Constraints |
|---|---|---|---|
| S1 | MC | ... | ... |
| ... | | | |

Table 4: The Question Table

| Session | Type | Level | URL |
|---|---|---|---|
| S1 | FIB | Difficult | ... |
| S1 | FIB | Easy | ... |
| ... | | | |

A tree traversal algorithm can be used to generate the structure pattern table. The database definition language program is fixed. But, the algorithm needs to generate the manipulation language program, which asserts the structure to the table. The other tables are constructed similarly.

## 5 Conclusions

We only presented the first part of a virtual university instruction system. The system aims to provide an integrated environment which allows instruction design, delivery, as well as assessment. The proposed course pattern approach to Web course design is just a beginning. We believe that, it is important to have a generic user interface of distance learning and virtual university operations. The interface has a fixed *interaction pattern*, which serves as the user navigation and interaction protocol of distance learning materials. The Web course pattern proposed in this paper is an internal design of such a methodology. We hope that, with a common interface of virtual university, the administrator, the instructor, and the students can all benefit from the generic access protocol of distance learning.

## References

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley, 1995.

[2] Hein, T.L.; Irvine, S.E. "Assessment of Student Understanding Using On-line Discussion Groups". In Proceedings of the 28th Annual Frontiers in Education Conference, 1998, Volume: 1 , Page(s): 130 -135.

[3] Hmelo, C.E.; Lunken, E.Y.; Gramoll, K.; Yusuf, I. "Multimedia courseware for teaching dynamic concepts: assessment of student learning". In Proceedings of the 25th Annual Frontiers in Education Conference, 1995, Volume: 1 , 1995 , Page(s): 2b2.19 - 2b2.23 vol.1

## A An Example of Structure Pattern Derivation

```
CoursePattern
:= StructurePattern
:= RCU StructureP
:= StructureP
:= TreePattern
:= Session 3< TreeP >
:= Session 3< StructurePattern StructurePattern StructurePattern >
:= Session 3< StructureP StructureP RCU StructureP >
:= Session 3< TreePattern TreePattern CourseUnitName @ TreePattern >
:= Session 3< Session 2< TreeP >
              Session 2< TreeP >
              CourseUnitName @ Session 2< TreeP > >
:= Session 3< Session 2< ListPattern ListPattern >
              Session 2< ListPattern Session >
              CourseUnitName @ Session 2< ListPattern ListPattern > >
:= Session 3< Session 2< 3[ ListP ] 2[ ListP ] >
              Session 2< Session 3[ ListP ] >
              CourseUnitName @ Session 2< 1[ ListP ] 2[ ListP ] > >
:= Session 3< Session 2< 3[ Session Session Session ] 2[ Session Session ] >
              Session 2< Session 3[ Session Session Session ] >
              CourseUnitName @ Session 2< 1[ Session ]
                                          2[ StructurePattern StructurePattern ] > >
:= Session 3< Session 2< 3[ Session Session Session ] 2[ Session Session ] >
              Session 2< Session 3[ Session Session Session ] >
              CourseUnitName @ Session 2< 1[ Session ]
                                          2[ RCU StructureP Session ] > >
:= Session 3< Session 2< 3[ Session Session Session ] 2[ Session Session ] >
              Session 2< Session 3[ Session Session Session ] >
              CourseUnitName @ Session 2< 1[ Session ]
                                          2[ CourseUnitName @ Session Session ] > >
```

## B An Example of Content Pattern Derivation

```
Session
:= SessionName SessionContent
     Pop-upQuizzes SessionQuestions
:= "Introduction to Linked List"
   WebDocument
   MultipleChoiceQuestion Constraints
   FillInBlankQuestion FillInBlankQuestion
:= "Introduction to Linked List"
   WebDocument
   MultipleChoiceQuestion Constraints or Constraints
   FillInBlankQuestion FillInBlankQuestion
:= "Introduction to Linked List"
   WebDocument
   MultipleChoiceQuestion TimeTrigger
                          or SpecificObjectMetTriger
   FillInBlankQuestion FillInBlankQuestion
:= "Introduction to Linked List"
   WebDocument
   MultipleChoiceQuestion TimeVisited >= Integer
                          or CurrentMObject = MObject
   FillInBlankQuestion FillInBlankQuestion
```