Automatic Trap Detection: A Debugging Mechanism for Abnormal Specification in the IMS Sequencing Controls

Freya H. Lin and Timothy K. Shih, Senior Member, IEEE

Abstract—Adaptive learning is a challenging trend of distance education. In order to adapt diversified teaching in accordance with individual abilities, educators emphasize on constructing personalized courseware as an Internet-based application. To cope with the drawback of nonsupervised blind spots on the dynamic navigation of learners, the SCORM 2004 specification adds the concept of sequences (i.e., based on IMS Simple Sequence Specification) into its content delivery. Sequencing rules describe how individual navigation is invoked, what learning objects are delivered during the navigation process, and the potential outputs of the process. However, there is a potential leak from using sequence specification. It is possible for an instruction designer to make a mistake similar to that made by any programmer in software development. In other words, a sequencing trap results from improper sequence composition, which may block learning activities from a normal navigation. Although the problem can be manually checked with an ordinary SCORM presentation tool, nonsystematic debugging can be an unsound solution and a time consuming job, since the complexity of courseware is usually proportional to the occurrences of traps. We point out the sequencing trap problem and its solution by using a fully automatic mechanism for trap detection. A Petri net is used in the computation of sequencing traps. The contribution of this paper can be used as a first technique for automatic sequencing testing in the IMS Simple Sequence Specification and SCORM 2004.

Index Terms—IMS Simple Sequence Specification, SCORM 2004, Petri net, automatic testing, distance learning, Web-based learning, sequence debugging.

1 INTRODUCTION

E-learning is convenient and important to students and teachers. Since the 1960s, CBT (Computer-Based Training) has provided an alternative teaching method to enhance traditional in-classroom learning. Nowadays, most instructors use conventional tools, such as Microsoft PowerPoint and Home Page editing tools, to create their learning content on the Internet. In the e-learning literature, there exist many methods and systems to create learning content. However, it is hard for instructors to exchange and reuse their learning materials unless a standard is enforced. There exist many standard proposals [10], [11], [26], [31] for asynchronous e-learning. But, too many standards will not solve the problem.

In 1997, the Advanced Distributed Learning (ADL) Initiative [30] proposed the Sharable Content Object Reference Model (SCORM) specification. SCORM, as a de facto model adopted in distance learning nowadays, provides durability, interoperability, accessibility, and reusability. SCORM Sequence and Navigation [1] was adopted from the IMS Simple Sequence Specification [12] to describe how sequencing behaviors are applied to track an individual learner’s progress and to enable adaptive (or personalized) learning mechanisms. Adaptive learning means the learners need the course “of their own”—at their pace and on their demand. Even with the same set of learning objects packed in a SCORM-compliant courseware, an individual needs diverse learning paths that take part in the same package. To facilitate personalized learning, which may provide another dimension toward the success of distance education on the Internet, Sequence and Navigation play a very important role [18].

There are research projects [15], [17] on transferring content structure to conform to the IMS Simple Sequence Specification. Other contributions [4], [13], [23], [33] provide environments and systems to realize SCORM sequencing behaviors. In addition, in [29], a method is proposed to navigate a learner appropriately by minimum rounds in a course according to the analysis of exam scores. Each learner can obtain his/her proper learning path and can achieve lesson goals in a short time. The above papers have contributions from the perspectives of authoring and delivering contents adaptively. It deserves mention that the structure representation of a SCORM-compliant courseware is in a XML file. It is too complex to implement a sequence specification if no suitable tool is provided. To ease SCORM course design, some authoring tools [14], [22], [32] allow users to edit sequencing rules by using ComboBox selection. Others [28] propose sequence templates to facilitate sequencing rules. After editing the rules, designers can save the sequence templates for reuse [28]. Although most authoring tools [9], [19], [21], [22] offer graphical user
interfaces (GUIs) to create SCORM-compliant courseware, the entire topology of a completed courseware is hard to image due to its complexity. Thus, a hypermedia authoring tool [3] and Petri net models [7], [8], [16], [34] are applied in the visualization of concept or sequences. Aha! (Adaptive Hypermedia Architecture) [3] is a well-known Adaptive Educational Hypermedia [5], [6], [20], [24] system that allows designers to create courses, concepts, concept relationships [2], and exams easily by a look-and-feel friendly interface and by making a translation from constructs to adaptive rules [25] automatically. In addition, Huang et al. [34] propose an automatic process during content creation to enable lecturers to efficiently generate courses that not only correspond to the SCORM standard but also arrange adaptive auxiliary materials dynamically. In [16], the High Level Petri Nets (HLPN) model is proposed to construct SCORM-compliant courseware by drag-and-drop graphic components of HLPN in each learning activity structure. However, HLPN cannot work well when a learning activity is applied to various sequence rules simultaneously. Furthermore, SCORM sequencing behaviors may not be intuitive. Content developers should also focus on interactive relationships among learning activities in addition to the contents. Nevertheless, in [7], [8], authoring tools can generate a Petri net graph automatically according to the sequences of each interactive learning activity. These solutions provide an efficient way to ease the understanding of navigation sequences.

However, the following abnormal situations will interrupt a learner’s activity due to improper usage of the IMS Simple Sequence Specification. These improper usages are not the problem of specification. It is a problem that could be created by the content designer.

- **Learning attempt is stuck in an activity.** Even though a learning activity is completed, the learner is blocked in the activity and cannot continue to pursue other learning activities. This situation can only be resolved by shutting down the LMS if the user uses the standard LMS provided by the ADL.
- **Learning attempt is fallen into a vicious circle.** A learning path results in a loop of a subactivity tree. Similarly, the ADL LMS needs to be terminated.
- **Deserved activities are not identified for delivery.** Sequencing rules cannot be fired. The situation forbids some Learning Objects to be presented.

The above situations that hinder learning activities are called sequencing traps. If rules defined by the content designer are not well considered, it is possible that a learner repeats the material in an endless cycle. Constructing rules appropriately for some educational purpose is an important topic of instruction design, but not the focus in this paper. In our approach, we look at the physical problem of user interactions. We can observe that sequence traps are created by the courseware designers. A similar situation may occur due to improper usage of programming languages (i.e., program bugs created by the programmer). Automatic testing tools were available in the software engineering literature. However, the literature of SCORM and e-learning has no solution to automatic trap detection so far. Extended from [7], the Sequencing-Oriented Petri Net (SOPN) model is used in this paper to facilitate the detecting of sequencing traps. Our approach can be summarized as the following:

1. Based on the Petri net model and the user-defined sequencing rules, our authoring tool generates an international representation of Petri net graph automatically. The representation is used in our detection algorithm. In addition, to illustrate the algorithm proposed, the visualized Petri net graph is used.

2. Two sequencing trap detecting methods are proposed based on tracing the Petri net graph. It is the first time that automatic testing of sequencing rules is implemented in a SCORM authoring tool.

Taking a new direction of study on sequencing behaviors, we focus on the trap detection of sequences. As a contribution of this paper, sequencing validation from the perspective of automatic software testing enables user-defined sequencing rules to become trustworthy. In addition, a traceable Petri net graph visualizes sequencing rules. In Section 2, we briefly introduce the Sequencing Definition Model and define the types of sequencing traps with illustrations of improper sequence composition. We also highlight the problem of detecting sequencing traps by using a Learning Management System (LMS) in a usual way. Section 3 proposes a Petri-net-based model to prove the existence of the problem and to further illustrate the traps. The contribution of our paper has its focus in Section 4, which provides a trap detection mechanism. The implementation of trap detection in a SCORM-2004-compliant authoring tool is presented in Section 5. Finally, our conclusions and future work are discussed in Section 6.

## 2 RELATED WORK

### 2.1 Sequencing Definition Model

The SCORM Sequence Specification is defined in [1] for the structure of learning activities, the sequencing strategy, and the application of behavior to external-system-triggered events. The IMS Simple Sequence Specification (as a part of SCORM 2004) provides a declarative and interoperable way to determine which learning object to present. Sequences can be divided into three main fundamentals: Activity Tree, Sequencing Definition Model, and SCORM Sequencing Behaviors. Readers are encouraged to read the SCORM specification to understand these terms before reading our proposed mechanism.

The SCORM specification is a well-written document. However, it is time consuming to understand the whole specification. Thus, in this section, we only discuss a few fundamental concepts for illustrating how our mechanism works. As previously mentioned, the Sequencing Definition Model has 12 parts. To easily probe into the patterns of instructional strategies, we use the categories of Foundation Construct and Operation Construct to illustrate the Sequencing Definition Model.

- **Foundation Constructs.** The Foundation Construct is the atomic structure able to control the sequencing behavior of a cluster. This category includes four elements of Sequencing Control Modes according to
the "Modules" and the "Exams" parts. The "Introduction to Computer Science" lecture can be completed only if the learner completes all learning materials, including "Introduction," three submodules (i.e., "Basic," "Operating Systems," and "Networks"), and "Exams." After completing the "Introduction," the learner can choose one of the three submodules, use the "Continue" button to jump to the next activity, or go back to a "Previous" activity (i.e., among "Basic," "Operating Systems," and "Networks"). If a Choice sequencing request is processed, the siblings are a valid target of the Choice request. After using an arbitrary selection or the "Continue" button, the learner should experience one of the "Pretests." The learner must attempt each question in the corresponding "Pretest" in a linear order. If the learner passes a "Pretest" (i.e., the correctness is higher than 60 percent), the module's learning objective has been satisfied and the module becomes disabled—the learner can ignore any of the rest of the lessons. If the learner fails to pass the exam, the learner is directed to the module of "Lessons" and, once completed, must take the "Posttest" module. Fig. 2 shows the corresponding complex sequencing setting on the activity tree in Fig. 1. Some sequencing setting can be applied to any activity in the Activity Tree (e.g., Choice Exit), but some will have no effect if applied to leaf activities (e.g., Flow). The deeper the height of an activity tree, the more complicated is the reciprocal setting for the course designer. In this example, there are two wrong settings that can result in sequencing traps.

If a learner determines his/her learning path as Introduction → Basic → Operating Systems → Networks → Exams, it is a workable learning path because the Flow is set to true of "Introduction to Computer Science" and, thus, "Modules" is fired. Under the intention of the course designer, the other learner can decide his/her learning path to be Introduction → Basic → Networks → Operating Systems → Exams. However, with an inadvertent improper setting, Choice Exit is set to false of "Networks." The learner will be blocked after he/she completes the "Networks" material. Such an improper setting should be detected and prevented in advance. In a practical situation without using our mechanism, it is hard to guarantee that no sequencing trap has been made by a course designer if the composition is very complicated.

To probe into improper sequencing composing results as sequencing traps, we give our analysis in Table 1, where all possible compositions of erroneous modes are listed. The analysis is tested with practical examples running on an LMS developed by ADL. The trap results were all verified [27]. The analysis is complete, as we use a truth table to eliminate all impossible combinations. Only possible combinations with or without traps are listed in Table 1.

Table 1 reveals two kinds of sequencing traps that can result in a learning attempt being stuck in Parent Activity or in Active Activity of Children. For instance, a Parent Activity has the Sequencing Control Flow and Sequencing Control Choice set to false (shown in Fig. 3). An LMS will disable the Continue and Previous navigation requests and no children will be valid targets for learners to choose. Learners will be stuck in the Parent Activity and a sequencing trap occurs. Cases 1, 2, 5, and 6 in Table 1 can be illustrated by Fig. 3.

<table>
<thead>
<tr>
<th>Introduction to Computer Science</th>
<th>Introduction Modules</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1 Test</td>
<td>Question 1, Question 2, Question 3</td>
<td></td>
</tr>
<tr>
<td>Module 1 Lessons</td>
<td>Computing History, Computer Architecture, Numbering Systems</td>
<td></td>
</tr>
<tr>
<td>Module 1 Posttest</td>
<td>Question 1, Question 2, Question 3</td>
<td></td>
</tr>
</tbody>
</table>

**Operating Systems**

- Module 2 Test: Question 1, Question 5
- Module 2 Lessons: Processes, Scheduling
- Module 2 Posttest: Question 1, Question 5

**Networks**

- Module 3 Test: Question 6, Question 7
- Module 3 Lessons: Network Protocols, Internet
- Module 3 Posttest: Question 6, Question 7

**Exams**

Fig. 1. A course example illustrated by an Activity Tree.
On the other hand, learners will be stuck in the active activity of children when there is no way to terminate or trigger another sibling. Once the Parent Activity has the Sequencing Control Choice set to true but the Sequencing Control Flow set to false, the children’s activities are all valid targets for learners to choose, but the LMS will not provide any mechanism for the learner to indicate their desire to "Continue" to the next activity or to go back to a "Previous" activity. When a learner’s attempt is at Activity 2 (shown in Fig. 4), the parent activity of Activity 2 has the Sequencing Control Choice defined as true and every sibling of Activity 2 is a valid target for a Choice navigation request. However, allowing one of Activity 1 or Activity 3 to be identified for delivery would result in the termination of Activity 2, violating the intention of Choice Exit control. At this moment, the learner cannot trigger Flow nor trigger Choice navigation requests. Thus, no activity could be identified for delivery and a sequencing trap occurs.

### TABLE 1
Composition of Sequencing Settings in Foundation Constructs

<table>
<thead>
<tr>
<th>Apply to</th>
<th>Parent Activity</th>
<th>Child(Active) Activity</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
<td>Flow</td>
<td>Forward Only</td>
<td>Choice</td>
</tr>
<tr>
<td>1</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>5</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>8</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>9</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>11</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>12</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>13</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>14</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>15</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>16</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Fig. 2. A corresponding sequencing to the Activity Tree in Fig. 1.

Fig. 3. An example of a sequencing trap on Parent Activity.
2.3 Detecting Sequencing Trap by Using LMS

To detect sequencing traps from improper sequencing settings, it is also possible for an author to manually find the trap by using an ordinary LMS (such as the one provided by ADL). However, the approach will be similar to testing a computer program using a brute force approach. Using an ordinary LMS will cause the following problems:

1. **It is hard to find all traps.** There are 16 combinations of using Sequencing Control Modes in a cluster. Quite a lot of testing steps need to be tried before the LMS can detect all traps. The LMS (e.g., ADL LMS) has the activity stuck in a trap. The user needs to restart the LMS and load the course again. This is very time consuming. In the case of using two or more levels of clusters, the combinations will be too complicated for testing using LMS.

2. **A sequencing trap can be confused with the ordinary ending of navigation.** In an ordinary LMS, in some stuck situations, it is not possible to tell if the activity is in a normal ending of navigation or in a trap. Thus, the user will be confused.

In [27], we had proposed a sequence testing function that uses a recursive method from a single cluster to a whole cluster tree by using a bottom-up method. However, this sequence testing function is only based on the truth table shown in Table 1. When the Sequencing Definition Model is fully implemented, the truth table will grow exponentially. As a result, the sequence testing function in [27] has a limitation on expansibility and flexibility. In this paper, we proposed a new mechanism which is discussed in the following sections. Using our trap detection mechanism and our authoring tool to debug traps, all sequencing traps will be found automatically in any limited depths of using clusters. The user will not be confused by telling the differences between a normal ending of navigation and a trap.

3 The Proposed Sequencing-Oriented Petri Net (SOPN)

Our new algorithm is based on a graph-based structure with control characteristics. In the literature, a few graph models, such as concept map, decision graph, and some concurrent processing modes, are known. In addition, Petri net is also available. We choose Petri net due to its publicity and ease of understanding, especially when we visually explain the problem of sequencing traps. However, in our authoring tool, it is not required for the users to use Petri net. Instead, an international representation of Petri net is generated automatically for the algorithm to detect traps.

Based on [7], the Petri net model is extended to have the ability of preventing and detecting sequencing traps. There are three reasons for using Petri net:

- **Graphic characteristic for visualization.** The graphic characteristic of Petri net supports a visible sequencing flow for the convenience of understanding.
- **Formal analysis.** Petri net is a formal process modeling technique with a precise definition to support workflow management. The technique was also proven useful in concurrent processing, distributed simulation, and multimedia presentation. Petri net is suitable to model the sequencing behavior of SCORM.
- **Expansibility and hierarchy.** The Petri net model can be extended to satisfy all possible combinations of sequencing rules. Moreover, subnet-oriented decomposition allows subcourses to be combined easily to decrease the complexity of computation.

### 3.1 Petri Net Model Revised

The revised Sequencing-Oriented Petri Net model (SOPN) is defined as a directed graph $PN = (P, T, F, W, M_0)$, where

1. $P = \{p_1, p_2, \ldots, p_m\} \cup \{c_1, c_2, \ldots, c_r\}$ is a finite set of places that consist of two subsets, the Ordinary Place subset (circle) and the Control Place subset (double circle), respectively. In the Ordinary Place subset, $\{p_1, p_2, \ldots, p_m\}$, each $p_i$ with the form of $(p_i, \{p_j, p_k, \ldots\})$, $p_i$ is the parent place and the set $\{p_j, p_k, \ldots\}$ represents the subnet (children) of $p_i$.

2. $T = \{t_{12}, t_{13}, \ldots, t_{ij}, \ldots t_{kl}\}$ is a finite set of transitions that draw by bars. The transition $t_{ij}$ implies an output transition to the place $p_i$ and an input transition to the place $p_j$.

3. $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs consisting of pairs $(p_i, t_{ij})$ or $(t_{ij}, p_i)$.

4. $W: F \to N$ is a weight function, where $N = \{1, 2, \ldots\}$ represents a set of positive integers. A $k$ weight arc can be interpreted as there being $k$ parallel arcs.

5. $M_0: P \to I$ is the initial marking (dot) function which assigns color tokens to each place in the net, where $I = \{0, 1, 2, \ldots\}$ represents a set of non-negative integers.

In the Ordinary Place subset $\{p_1, p_2, \ldots, p_m\}$, if $p_i$ with the form of $(p_i, \phi)$, it represents that $p_i$ is not only a child place in the cluster but also a leaf activity in the whole Activity Tree. Such a place may be considered as learning material including lessons, assessments, or courses. In contrast, if $p_i$ has the form of $(p_i, \{p_j, p_k, \ldots\})$, it represents that $p_i$ is a parent place in the cluster. Although the parent place stands for an abstract activity without any physical learning object mapping, it plays an important role...
in setting sequencing rules and showing the Activity Tree. Another type of place is the Control Place, which gives assistance to model management. In other words, SOPN utilizes the Control Place to illustrate the Operation Constructs, such as attempt limit, skip, and so on. T is a finite set of transitions, which can be an event, computation step, or state changing operator. The arc directs the information flow outgoing from input place to either a point of departure or a temporary pause. The dynamic behavior of the model simulates the firing rules. A transition is enabled when the token number of its input place is greater than the weight of its input arc. A transition must be enabled in order to fire. If the transition is fired, the token of the input place will be moved to the output place according to the weight of the output arc.

3.2 Applying SOPN on Foundation Constructs

As mentioned in Section 2, Foundation Constructs are atomic structure to control the sequencing behavior of a cluster. Sequencing Control Flow, Sequencing Control Forward Only, and Sequencing Control Choice are applied to parent activity and have effect to the child activities. Sequencing Control Choice Exit is applied to any active activity to conditionally constrain valid targets of Choice navigation requests. First, we apply SOPN on these four modes of sequencing setting.

When a Parent Activity has a Sequencing Control Flow defined as true, Fig. 5b shows how flow construction is applied by the SOPN. The directed graph is defined by the quintuple

\[
P_{\text{Flow}} = (P_{\text{Flow}}, T_{\text{Flow}}, F_{\text{Flow}}, W_{\text{Flow}}, M_{\text{Flow}}),
\]

where

\[
P_{\text{Flow}} = \{p_1, p_2, p_3, p_4\},
\]

\[
T_{\text{Flow}} = \{t_{12}, t_{23}, t_{32}, t_{34}, t_{41}\},
\]

\[
F_{\text{Flow}} = \{(p_1, p_2), (p_2, t_{23}), (p_2, t_{32}), (p_3, t_{34}), (p_4, t_{41}), (p_4, t_{43}), (t_{12}, p_2), (t_{23}, p_3), (t_{32}, p_2), (t_{34}, p_4), (t_{43}, p_3), (t_{41}, p_1)\};
\]

\[
W_{\text{Flow}} : (p_1, t_{12}) = 1, (p_1, t_{23}) = 1, (p_1, t_{32}) = 1, (p_1, t_{34}) = 1,
\]

\[
(p_2, t_{12}) = 1, (p_2, t_{23}) = 1, (p_2, t_{32}) = 1, (p_2, t_{43}) = 1,
\]

\[
(t_{12}, p_2) = 1, (t_{23}, p_3) = 1, (t_{32}, p_2) = 1, (t_{34}, p_4) = 1, (t_{43}, p_3) = 1, (t_{41}, p_1) = 1;
\]

\[
M_{\text{Flow}} : p_1 = 1, p_2 = 0, p_3 = 0, p_4 = 0.
\]

When a parent activity has Sequencing Control Forward Only defined as true, the construction applying by SOPN (as shown in Fig. 5d) is similar to Sequencing Control Flow. The difference is that “Previous” activity is forbidden among child activities. As a result, the place set and initial marking function will be the same but the transitions and arcs will be changed as follows:

\[
T_{\text{Only}} = \{t_{12}, t_{23}, t_{34}, t_{41}\};
\]

\[
F_{\text{Only}} = \{(p_1, t_{12}), (p_2, t_{23}), (p_3, t_{34}), (p_4, t_{41}), (t_{12}, p_2),
\]

\[
(t_{23}, p_3), (t_{34}, p_4), (t_{41}, p_1)\};
\]

Notice that learners should start their learning path from parent place \( p_1 \) (abstract activity) when the sequencing rule is firing and terminate the learning activity at parent place \( p_1 \) (abstract activity) when the objective is satisfied.

The Sequencing Control Choice represents that the learner can select lessons in any order. While parent
place $p_1$ has a token (as shown in Fig. 5f), transitions $t_{12}, t_{13}, t_{14}$ are enabled to be fired. If $t_{13}$ fires, $t_{13}$ removes the enabling token from place $p_1$ and puts tokens in $p_1$ to show that Lesson 2 is an active activity.

Meanwhile, the Sequencing Control Choice Exit can be applied to active activity and decide if siblings of this activity are a valid target of a Choice navigation request. When the Sequencing Control Choice Exit is defined as true, the learner has the ability to trigger Choice navigation requests that target nondescendent activities. To simulate such navigation, Fig. 6 illustrates the example that the Sequencing Control Choice Exit is set to true for the active activity (Lesson 3). The output transition $t_{ap}$ will be added to the place $p_a$. Thus, there are directed arcs connecting place $p_a$ to transition $t_{ap}$ and transitions to places $p_1$, which represents the parent activity. In light of the above development, Fig. 6b presents Choice Exit construction. By firing transition $t_{ap}$, the token is removed from $p_a$ to $p_p$ so that $p_2$ and $p_3$ are valid targets of Choice navigation requests within $p_p$. On the contrary, if the active activity is $p_{14}$ due to the fact that the value of the Sequencing Control Choice Exit is false, the siblings of $p_3$ are invalid targets of a Choice navigation request for the parent place.

### 3.3 Applying SOPN on Operation Constructs

As mentioned in Section 2, Foundation Constructs are atomic structures to control the sequencing behavior of a cluster. Sequencing Control Flow, Sequencing Control Forward Only, and Sequencing Control Choice are applied to a parent activity and have an effect on the child activities. The Sequencing Control Choice Exit is applied to any active activity to conditionally constrain valid targets of Choice navigation requests. Firstly, we apply SOPN on these four modes of sequencing setting.

In conjunction with Foundation Constructs, Operation Constructs, such as Skip, Limit Condition, or Rollup, can be modeled by control places of SOPN. Skip is one kind of action in the if-then action sequencing rule (i.e., similar to the if-then statement in ordinary programming languages). The Limit Condition is based on the Tracking Model (part of the Sequencing and Navigation specification). Limited conditions can override Sequencing Rules. Rollup is the process of evaluating the Objective and Attempt Progress data for a set of child activities with respect to a parent activity. Objective and Attempt Progress allows an instructor designer to set an explicit objective of learning and the status of learner's progress. This section illustrates how SOPN works on Operation Constructs.

The Limit Condition describes conditions under which an activity is not allowed to be delivered. Since SCORM 2004 does not restrict using any time-based limit conditions (i.e., how long the user should read), our model only focuses on the maximum number of attempts for the activity (i.e., the Limit Condition Attempt Limit). Adding control places to the input transitions of this limited learning activity can control the number of attempts. According to the number of tokens in the control place, we set a limit on how many times the learning material can be read. As illustrated in Fig. 7, if the number of tokens in $cp_1$ is equals two, the learner can visit “Lesson2” twice.

Skip is the action when a learner wants to ignore some learning materials. We utilize control places and skip-event transitions to allow learners to either directly traverse the content in a linear manner or skip one of the learning materials. Based on a Foundation Construct (i.e., the Sequencing Control Flow Only is set to True in Fig. 5c), we use an operation construct, [If condition satisfied then skip], which is applied to Lesson2 in Fig. 8. The combination indicates that, after completing Lesson 1, a condition is tested (i.e., a quiz or an exam). The condition can be modeled by using a subnet concept of Petri net. The consequence from this subnet allows $cp_1$ and $cp_2$ control. If the condition is satisfied, the token will be added to control place $cp_1$. Otherwise, the other control place $cp_2$ will contain the token. In Fig. 8, the double circles represent these control places. If the condition is satisfied (e.g., a learner passes the exam), both input places ($p_2$ and $cp_1$) of the skip-event transition ($t_{23}$) contain tokens, and $t_{23}$ can be fired. The activity Lesson2 is skipped and the activity Lesson3 is then traversed.
Rollup is the process of evaluating Objective and Attempt Progress data for a set of child activities with respect to a parent activity. The concept can be modeled by the hierarchical feature of Petri nets. An entire activity tree can be replaced by a single place or transition and vice versa. In our model, we use a token number and weight of arc to determine whether Rollup is satisfied. Following the example illustrated in Fig. 8, as a new example in Fig. 9, after Lesson1 is completed, the subnet is invoked. Within the subnet, two places called Pretest and Posttest are used. If a learner passes the Pretest, it is not necessary for the learner to visit the Remedial Course. Otherwise, the Remedial Course is visited and a Posttest is used.

3.4 Applying SOPN on Elements Composition in the Sequencing Definition Model

By applying SOPN to an individual element of the Sequencing Control Modes, the atomic structure is built to control the sequencing behavior of a cluster. Based on those atomic structures, various compositions of these structures or other elements in the SCORM Sequencing Definition Model are extended. The extended methods depend on the relation between those composed elements. If two composing elements have an inclusive relationship, the extended method will adopt the Intersection operation. Otherwise, the Union operation will be adopted to combine existing Petri nets.

Intersection Operation

There are two directed graphs, $PN_1 = (P_1, T_1, F_1, W_1, M_{01})$ and $PN_2 = (P_2, T_2, F_2, W_2, M_{02})$, to model their own sequencing behavior. We construct from $PN_1$ and $PN_2$ an intersection Petri net $PN^* = (P^*, T^*, F^*, W^*, M_{0}^*)$, whose place set $P^* = P_1 \cap P_2$, transition set $T^* = T_1 \cap T_2$, and arc set $F^* = F_1 \cap F_2$.

Weight function $W^*: F^* \rightarrow Min(N_a, N_b)$ whenever $W_1 : (p_i, t_{ij}) = N_a$ and $W_2 : (p_i, t_{ij}) = N_b$
Or whenever $W_1 : (t_{ji}, p_i) = N_a$ and $W_2 : (t_{ji}, p_i) = N_b$

Initial marking function $M_{0}^*: p_i = Min(I_a, I_b), p_i \in P^*$ whenever $M_{01}: p_i = I_a$ and $M_{02}: p_i = I_b$

Union Operation

There are two directed graphs, $PN_1 = (P_1, T_1, F_1, W_1, M_{01})$ and $PN_2 = (P_2, T_2, F_2, W_2, M_{02})$, to model their own sequencing behavior. We construct from $PN_1$ and $PN_2$ a union Petri net $PN_U = (P_U, T_U, F_U, W_U, M_{0})$, whose place set $P_U = P_1 \cup P_2$, transition set $T_U = T_1 \cup T_2$, and arc set $F_U = F_1 \cup F_2$.

Weight function $W_U: F_U \rightarrow Max(N_a, N_b)$ whenever $W_1 : (p_i, t_{ij}) = N_a$ and $W_2 : (p_i, t_{ij}) = N_b$
Or whenever $W_1 : (t_{ji}, p_i) = N_a$ and $W_2 : (t_{ji}, p_i) = N_b$

Initial marking function $M_{0}: p_i = Max(I_a, I_b), p_i \in P_U$ whenever $M_{01}: p_i = I_a$ and $M_{02}: p_i = I_b$

Through our Sequencing-Oriented Petri Net model, each cluster has its ability to illustrate the composing elements of
4 SEQUENCING TRAP DETECTION

4.1 Detect by Petri Net Graph

Whether a learning activity is stuck in an activity or fallen into a vicious learning circle, a sequencing trap shows that the learners are mired in learning difficulties. According to the setting of the Sequencing Control Mode, a Petri net graph can be built. When a composition fails to build such a graph, a sequencing trap on Parent Activity occurs. Taking an example from Fig. 3, Parent Activity has both the Sequencing Control Flow and the Sequencing Control Choice set to false. It is impossible to create any input arc or output arc. In other words, whether a sequencing trap can be prevented depends on whether a Petri net graph can be built.

4.2 Detect by Flow Matrix

Petri net provides a visible and traceable way for SCORM Sequencing. If sequence composing cannot be illustrated by Petri net, then there is a sequencing trap. On the contrary, the existence of Petri net does not imply avoidance of the sequencing trap. In order to solve this problem, we need a detection algorithm. This algorithm not only finds traps in foundation constructs, but also traps in operation constructs.

From Fig. 10a, we cite the Limit Condition Attempt Limit (see the SCORM 2004 specification) as an instance. In a practical learning experience, it is possible that a content developer wants to limit the number of attempts that a learner is permitted on a given learning activity. Adding a control place to the input transitions of this limited learning activity can control the number of attempts. According to the number of tokens in the Control Place, we limit how many times the learning material can be read. For example, in Fig. 10a, $P_3$ has a Limit Condition Attempt Limit defined as 2. That is, the learner is permitted to learn $P_3$ only twice. As shown in Fig. 10b, $cp_1$ is used, which has two tokens to limit the firing time of input transition $t_{13}, t_{23}, t_{43}$.

Initially, we construct a Flow Matrix, $M$, to keep track of whether there exists a directed edge from place $i$ to place $j$, where $i \neq j$, and $i$ includes the Control Place. From Fig. 10b, the following Flow Matrix is constructed:

$$
M = \begin{bmatrix}
1 & 0 & 1 & 1 & 0 \\
2 & 0 & 1 & 0 & 0 \\
3 & 0 & 1 & 0 & 1 \\
4 & 1 & 0 & 1 & 0 \\
cp & 0 & 2 & 0 & 0
\end{bmatrix}
$$

Usually, the root activity of an Activity Tree is encoded as number “1” and in turns for other activities. Within a token, $P_1$ is a start point and $P_1$ randomly picks one transition to be fired. If the selected place has the Limit Condition, we further check whether the value of the Control Place at the selected place is greater than one. If it is, we decrease the value of the Control Place by one. Otherwise, it means the learner cannot exit from this limited place to others. In other words, the sequence is trapped. Repeat the above selecting process until the selection remains the root activity or we cannot find any place to advance the selected process. If the selection is the root activity, we find a workable path; otherwise, the sequence is trapped.

We illustrate a trap and a workable path as a legend. The trapped path is $P_1 \rightarrow P_3 \rightarrow P_2 \rightarrow P_4 \rightarrow P_1$; if we arrive at $P_2$ twice, we cannot find any place to make process and the sequence falls in a trap. We show the process of the Flow Matrix in Fig. 11. If we choose the path as $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_1$, we find a workable path. We show the change of the Flow Matrix in Fig. 11 continued.
Fig. 11. Trap path of the flow matrix.
Fig. 11 continued. Trap path of the flow matrix.
4.3 The Proposed Algorithms

In the last section, we explain how a Petri net can be used as a computation foundation to detect traps. As a contribution of this paper, we propose the detection algorithm in this section. The algorithm is implemented in our authoring tool, which is discussed in the next section. The algorithm follows:

//Detection Algorithm for a cluster
//Input: An applied Petri net $PN = (P, T, F, W, M_0)$ for a cluster
//Output: Sequencing trap occurs in a cluster

Begin
1. Construct a Flow Matrix, $M$

$$m_{i,j} = \begin{cases} 
1 & \text{if exists a direct edge from } i \text{ to } j, \\ 0 & \text{otherwise} 
\end{cases} \quad 1 \leq i, j \leq n,$$

where $n$ is the number of places, and we add one row for Control Place by setting its value with available counts.

2. Set the parent place as the starting place $i$

3. Randomly select a connected place $j$ in $M$, where

$$m_{i,j} = 1$$

3.1 If place $j$ is a place with Limit Condition, check the value of $m_{cp,j}$
   
   i. If the value is greater than one, decrease $m_{cp,j}$ by one.
   
   ii. Else sequence is trapped.

4. If place $j$ selected in step 3 is the root, we have found a workable path.

If we cannot find any place for making process, sequence is trapped, signal an error message.

Otherwise, repeat step 2 to 3, until all connected places from place $i$ are selected.

End

As we have mentioned before, activities are placed within an Activity Tree that consists of clusters. A cluster includes a single parent activity and its immediate child activities. Sets of rules are associated with each activity and its cluster. An abstract module gathers learning objects, lessons or questions, or its sub-abstract-model to represent the activity tree. The concept can be mapped onto the feature of Petri net to construct a framework. An entire activity tree may be replaced by a single place or transition for modeling at a more abstract level or places and transitions may be replaced by subnets to provide more detailed modeling. For such a reason, extending the above Detection Algorithm to a more complex tree structure is necessary to detect whether sequencing traps exist in the whole course structure. The checking algorithm is presented below. Recursion is used to detect the complex tree. The strategy is bottom-up by starting from a subtree with all leaves children and the height of this subtree is equal to two. We also check this subtree by the previous Detection Algorithm. If a workable path exists, the checking process will not stop until all subtrees are examined or the root node is arrived at. If there is a sequencing trap in any one of the subtrees, we signal an error message.

//Checking Algorithm
//Input: Whole course structure and Petri nets for each cluster
//Output: Sequencing Traps occurs with error messages

Begin
1. Starting from a subtree $T_{sub}$, with the form of $(p_i, \{p_{i+1}, \ldots, p_j\})$ and the height of $T_{sub}$ is 2, where $p_i$ is the root of this subtree and $p_{i+1}, \ldots, p_j$ are all leaves children.

2. If we can find a workable path in Detection Algorithm of $T_{sub}$, repeat step 1 until we examine all subtrees or $p_i$ in step 1 is the root of the whole tree.

Else
we concluded that we cannot find a workable path of the whole tree.

End

5 Implementation of a Trap Detection Tool

The trap detection mechanism is implemented in the Hard SCORM Authoring Tool, which is also a content aggregation and packaging tool for SCORM 2004. The IMS Simple Sequence Specification is adopted in the SCORM 2004 specification. Sequencing and navigation information can be stored in the course by specifying sequencing rules in the XML format. However, editing XML-based sequence rules is hard for ordinary instructors due to the complexity of XML syntax. An effortless toolkit, Sequencing Editor, for instructors to specify sequence rules is implemented in our authoring tool. In the Sequencing Editor, instructors can simply set sequencing rules by using the drop-down menu and fill in the information for the specific data value. The schema of the data value items is stored in a database for the conceivable extension in the SCORM specification.

Upon using the Sequencing Editor in our tool and a tree structure for representing the content aggregation, the SOPN is generated. Based on our proposed sequencing trap detecting mechanism, different colors shown in the aggregation tree indicate different types of errors. The authors should fix these errors themselves before launching the content package for public usage.

The authoring tool is available for the industry and universities. We allow readers to download and use the tool freely for noncommercial purposes. The tool is available at the ADL Website (see [30], under the category of “third party tools”). Currently, more than 2,600 users have visited the tool Website and there are over 20 universities and industry organizations around the world using the authoring tool.

As compared to an authoring tool called RELOAD (http://www.reload.ac.uk), developed by the University of Bolton and the University of Strathclyde, as well as by ADL [30], our authoring tool provides a similar functionality in creating SCORM courses. However, in a typical editing section, the creator using RELOAD could use the official LMS offered by ADL to check the soundness of content package created. However, it is time consuming to
jump between two tools (i.e., editor and LMS). In some cases, if a trap is found, the author needs to quit the LMS before the next course can be tested. Our Hard SCORM Authoring Tool, on the other hand, provides a seamless integration of editing and trap detection. Our authoring tool was demonstrated in several official meetings at ADL and received a lot of good comments.

6 Conclusion

Distance education is one of the most important impacts to Internet and Web-based technologies. Among Web-based learning, adaptive learning is considered a frontier research direction, where sequencing and navigation play a key role. In accordance with this vision, the IMS Simple Sequence Specification was adopted as part of the SCORM 2004. Our main contribution to the specification is to provide a fully automatic mechanism to find traps among different learning objects in a SCORM-based courseware. The proposed Sequencing-Oriented Petri Net is applied on the mechanism to justify the operational behavior of sequences. Through the implementation of the Intersection or Union operation on Petri net combination, all possible compositions of learning objects in the Sequencing Definition Mode of SCORM can be represented. Our proposed Sequencing-Oriented Petri Net Model not only visualizes sequencing topology easily but can also detect a sequencing trap in advance. Starting from the bottom to the top of a course hierarchy, the Checking Algorithm and the Detection Algorithm are implemented and used in our authoring tool.

The automatic detection of a sequencing trap can assist a course designer in fixing error sequences. The algorithm implemented in the authoring tool was demonstrated on the Advanced Distributed Learning Initiative in one of the SCORM exposition conferences. We received a very high recommendation from the audiences. To our understanding, this is the first time that sequencing and navigation topology can be automatically tested. We consider this as the main contribution of our research project to build SCORM-compliant software systems. For interested readers, the authoring tool is available at the Website of ADL as well as at http://www.mine.tku.edu.tw/scorm.

Currently, we are developing our trap detection tool using Java (the original was implemented in C++) and we plan to provide application program interface for public usage. Yet, our trap detection mechanism has no capability to summarize commonly occurring mistakes from the users. Our future work is to build a tool to analyze these mistakes and to suggest to ADL or IMS to refine the Simple Sequence Specification in order to avoid user errors.

Acknowledgments

This research was supported in part by the National Science Council of Taiwan, ROC, under grants NSC 94-2218-E-011-011 and NSC 96-2511-H-003-013.

References

Freya H. Lin received the MS degree from the Department of Informatics at Fo Guang University, Taiwan, in 2003 and the PhD degree from the Department of Computer Science and Information Engineering of Tamkang University, Taiwan, in 2007. She is currently an assistant professor in the Department of Information Management at the Chihlee Institute of Technology, Taiwan. Her research interests include adaptive learning, e-learning, Petri net, and data mining.

Timothy K. Shih is a professor in the Department of Computer Science at the National Taipei University of Education, Taiwan. As a senior member of the IEEE, he sits on the Educational Activities Board of the IEEE Computer Society. His current research interests include multimedia computing and distance learning. He was the founder and co-editor-in-chief of the International Journal of Distance Education Technologies (Idea Group Publishing). He is the associate editor of the ACM Transactions on Internet Technology and the associate editor of the IEEE Transactions on Learning Technologies. He was also the associate editor of the IEEE Transactions on Multimedia. He is a fellow of the Institution of Engineering and Technology (IET) and a member of ACM.