Knowledge Modeling for Developing Program Planning Agents

Fuhua Lin, Qin Li, Dunwei Wen
School of Computing and Information Systems, Athabasca University
Athabasca, AB, Canada, T9S 3A3
Tel: 1-780-4212548, Fax: 1-780-6756186
Email: [oscarl | qinl | dunweiw]@athabascau.ca

Abstract: This paper describes a method of domain knowledge modeling for program planning and scheduling in intelligent e-Learning advising systems, focusing on the modeling and representation of precedence relations among course learning objects encoded in model curricular and the representation of domain experts' knowledge using Petri nets formalism and a XML-based markup language. We developed a Web-based program model editor.

Keywords: E-Learning & innovations in teaching, knowledge modeling and representation, program planning, Petri nets.

I. Introduction

Curriculum planning for student advising is an interactive decision-making process, which assist learners in the development of meaningful learning plans which are compatible with their career/life goals maximize educational potential through communication and information exchanges with an educator. An intelligent e-learning planning agent is a knowledge-based system able to assist e-learners to generate a personalized curriculum best suitable for them. The abilities of such a system hinge on possessing knowledge about the domain and program structure, learners, and pedagogy.

One of the important tasks in developing intelligent program planning systems is to model and represent the program model in the context so that the scheduling agents can use it to generate learning plans flexibly and adaptively. Due to the complex pre-requisite relationships among the course learning objects, this task is not trivial. This paper presents a Petri nets based model of program structure knowledge modeling and XML-based representation for dynamic program planning. We develop a knowledge editor for constructing and maintaining curriculum models.

To demonstrate the feasibility of the model, we developed an intelligent advisor able to generate personalized curriculum and generating a schedule for the courses selection according to students' temporal constraints, and course availability.

This paper mainly describes the method of domain knowledge modeling and representation for program planning and scheduling in e-Learning, focusing on the representation of precedence relations among course learning objects encoded in model curricular and the representation of domain experts' knowledge.

We proposed an extended Petri nets model to represent the logical relationships among courses in a curriculum. We developed a Web-based knowledge editor providing straightforward modification to accommodate addition or deletion of courses, prerequisites, co-requisites, and rules.

To demonstrate the feasibility of the model and the editor, we show a real-world example of Petri nets represented course-flow for MSc IS program of Athabasca University of Canada.

II. Literature Review

The literature records several efforts to employ computers to reduce the tedium, and improve the consistency, in student advising [1-7]. These systems exploit AI, expert systems or knowledge-based systems to provide some of the functions of a faculty advisor to students. Several of them provide students with a prioritized list of suggested courses and check prerequisites and sequencing. Most of them were designed to run on mainframes environments or standalone PCs. Gunadhi, et al., (1995) presented a framework for an intelligent advisory system for college students that combine object-oriented and knowledge-based paradigms [3]. They did not address how to acquire and incorporate "real-life" expertise and how to interface to other systems in the university such as online registration system, academic and student databases.

Flow diagrams, narrative descriptions, and time-line analysis are useful for a variety of modeling problems. Those tools however make it difficult or impossible to expose critical time-dependencies, task concurrencies, and event-driven behavior. Petri nets are a useful and powerful modeling tool and overcome the aforementioned shortcomings [8].

In curriculum modeling, an important task is representing the complex logical relationships among the prerequisites. Both state machines and graphs do not have capability to model curricular. State machines with the property that each transition has exactly one incoming and one outgoing arc or flow relation. They are a subclass of P/T nets [8]. State machines allow us to model choice or decision. Because each place may have multiple output transitions, they do not allow modeling of synchronization or concurrent activities, which are common in a curriculum. Concurrent activities require that several transitions be enabled concurrently [9]. In a marked graph, each place has

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 366 - 371.

only one incoming and one outgoing flow relation; thus, marked graphs do not allow modeling of choice while choices are important in course-choosing.

Knowledge modeling using Petri nets and its extensions in AI applications has received much attention. Net related models have been proposed for validating knowledge bases [10]; diagnostic knowledge representation and reasoning [11], determining requirements [12], and representing knowledge in discrete event systems [13].

K. Lee and M. Y. Jung (1994) presented a method for flexible operation planning using the Petri net approach [14]. A Petri net is used as a unified framework for both operation planning and plan representation.

III. Curriculum Petri Nets

Before describing the proposed Petri nets model, let us review the ordinary Place-Transition Nets.

III. 1 Place-Transition Nets (P/T nets)

Place-Transition nets (P/T nets) are the basic type of Petri nets from which all other types are derived. They consist of four elements [8]:

(1) **Places**: places model conditions or objects such as program variables. They are represented by circles.

(2) **Tokens**: tokens are represented by black dots. These are contained within places and represent the specific value of the condition or object which that place represents. The initial arrangement of tokens on places is known as the *initial marking* of the Petri net.

(3) **Transitions:** transitions are represented by hollow rectangles and model activities which change the values of conditions and objects.

(4) **Arcs**: arcs are represented by lines connecting places and transitions. These indicate which objects are changed by which activities. As Place-Transition nets are bipartite, arcs may only connect places to transitions or transitions to places, but not places to places or transitions to transitions. An arc may have a *weight*, which specifies how many tokens are created or destroyed when a transition to which it is attached is fired.

A P/T net can be defined formally using functional notation:

Definition 3.1: (P/T nets)

A Place-Transition net is a 5-tuple

$$PN = (P, T, I_-, I_+, M_0)$$

where

$P = \{p_1, \ldots, p_n\} (n < 0)$ is a finite and non-empty set of places;

$T = \{t_1, \ldots, t_m\} (m > 0)$ is a finite and non-empty set of transitions, $P \cap T = \Phi$;

$F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called flow relations;

$I_-, I_+ : P \times T \rightarrow \mathrm{N}_0$ are the backward and forward incidence functions respectively; If $I_-(p,t) > 0$, an arc leads from place p to transition t, whilst if $I_+(p,t) > 0$ then an arc leads from transition t to place p; $M_0 : P \rightarrow \mathrm{N}_0$ is the initial marking.

The dynamic behavior of a P/T net is determined by rules concerning the enabling and firing of transitions. When the arcs connecting a transition to its input places have a weight of one, the transition is enabled if all of its input places are marked with at least one token. Only an enabled transition may fire. When it does, one token on each of its input places is removed and one token is created on each of its output places. It is possible, however, for an arc to have a weight greater than one. In this case, if the arc is an input arc then the transition is only enabled if the number of tokens on the place to which it is connected is equal to or greater than its weight. When then transition is fired, the number of tokens removed on the place is equal to the arc's weight. If it is an output arc, firing the transition creates the number of tokens on the output place equivalent to the arc's weight. The numerical values of each of the I_- and I_+ functions correspond to the weights of the arcs connecting places and transitions. By convention arc weights of 1 are not shown explicitly.

The rules for the enabling and firing of transitions can be formalized thus:

Definition 3.2: (Firing rules of P/T nets)

If $PN = (P, T, I_-, I_+, M_0)$ is a P/T net. . A marking of a P/T net is a function $M : P \rightarrow \mathrm{N}_0$, where $M(p)$ is the number of tokens on place p, $\mathrm{N}_0 = \{0, 1, \ldots\}$;

A set $P_1 \subseteq P$ is marked at marking M, only and if only (iff) $\exists p \in P_1 : M(p) > 0$; otherwise P_1 is unmarked or empty at M; A transition t . T is enabled at M, denoted by $M[t>$, iff $M(p) \geq I_-(p,t), \forall p \in P$; A transition t . T, enabled at marking M, may fire yielding a new marking M' where

$M'(p) = M(p) - I_-(p,t) + I_+(p,t), \forall p \in P$,

denoted by $M[t > M'$. In this case M' is directly reachable from M and we write $M \rightarrow M'$. Let $\rightarrow^*$ be the reflexive and transitive closure of $\rightarrow$. A marking M' is reachable from M, iff $M \rightarrow^* M'$; A firing sequence of PN is a finite sequence of transitions $\sigma = t_1 \ldots t_n \quad n \geq 0$ such that there are markings $M_1, \ldots, M_{n+1}$ satisfying $M_i[t_i > M_{i+1}, \forall i = 1, \ldots, n$. A shorthand notation for this case is $M_1[\sigma >$ and $M_1[\sigma > M_{n+1}$, respectively. The empty firing sequence denoted by ε and $M[\varepsilon > M$ always holds.

The firing of a transition when the P/T net has one marking creates a new marking. The set of all markings

which are reachable from M_0 is known as the *reachability set* of the net and the connections between the markings in this set are represented by the reachability graph.

We extended P/T nets to model concurrent course taking activities, complex course dependency relations (e.g. pre-requisites,co-requisites), preferences (priorities), and exclusion.

III. 2 Curriculum P/T Nets

Course-taking process for an e-learner can be viewed as a process of changing the states of knowledge of the e-learner. To take a course, a learner's state of knowledge must meet the prerequisite requirement. After completing a course, the learner's state of knowledge is changed. The process of taking courses is then represented by a nondeterministic firing and passing the markers. The state of knowledge of a learner is thus denoted by the markup of the Petri net.

Definition 3.3 (Curriculum P/T nets):

A Curriculum P/T net

$$PN = (P, \quad T, \quad I_-, \quad I_+, \quad M_0)$$

is an extended P/T net,

$$PN = (P, \quad T, \quad I_-, \quad I_+, \quad M_0)$$

defined in **Definition 3.1** and **Definition 3.2** with the following extensions.

(1) Transitions *T*: Learning a course is mapped as a transition in a curriculum Petri Net, called course transition.
(2) Places *P*: We use places of P/T nets to indicate three types of the states in a program-perusing process.
(a) *Done Places*: There is only one output place for each course transition. That output place is named as Done Place. For example, the output place for course transition "Java Programming" is done place p_2 in Figure 1.
(b) *Ready Places*: We use an input place to represent the readiness of a prerequisite course of a course transition. There may be zero, one or many input places for a course transition. If a course has no prerequisite course, we still add an input place for that transition. That input places are names as Ready Places. If a done place of a course transition may be a ready place of another course transition. For example, course "Object-Oriented Programming (COMP501)" has no prerequisite courses in the program; its input place is Ready place p_1; p_2 is also a Ready place of course transition "Distributed Systems (COMP689)" as shown in Figure 1.

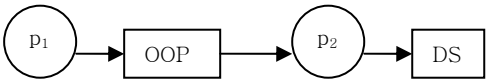

Figure 1: Example for Ready Places, Done places, and course transition.

(c) Choice places: we use a special place as a common input place of a set of course transitions to represent a situation of multiple choices in a curriculum. For example, an MSc IS student at Athabasca University can take only either "IS integration essay" or "IS Integration Project", once the student completes the IS Foundation Courses and Core Courses. So, we set a Choice place p_3 to represent the case of exclusion (see Figure 2).

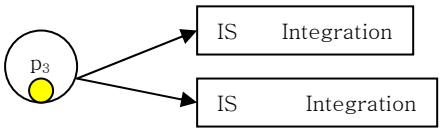

Figure 2: Example for choice places

(3) Markings M: The markings in an input place *p* of a course transition *t* represent the readiness of the prerequisite course t_0 in which the place *p* is its output place. If there is a token in an output place of a course transition, then the student already finishes the course.
(4) Incidence functions I_+, I_-: Considering one course can be the prerequisites of many courses, and it is hard to predict how many succeed courses of course *t* will be selected by a learner pursuing a program. Also, a learner may re-take a course to get a better mark. To keep the status that this course has been completed during the process of the program completion, we use a special incidence functions for all places in curriculum P/T nets:

- Once course *t* is done, one token will be created in its Done place *p*, i.e. $I_+(p,t) = 1$;
- No token in a Ready place *p* will be destroyed when its course transition *t* is fired, i.e. $I_-(p,t) = 0$;
- For a Choice place, one token will be removed from it when its course transition is fired, i.e. $I_-(p,t) = 1$.

(5) Firing rules: We extend the ordinary P/T nets by assigning a new firing rule:

A transition *t* in *T* is enabled at *M*, denoted by $M[t>, iff$ the condition predicate *pre*(*t*) holds: predicate *pre*(*t*) is formed by using the Boolean operators (AND, OR, NOT) over the place set $\{M(p): I_-(p,t) > 0, p \in P\}$.

The predicate only contains Boolean operators OR, AND, and NOT, and does not contain XOR operator, because the case of exclusion is represented by *Choice* places.

We classify the all possible pre-requisites into four categories:

Basic type:
(1) AND: $C_1 \wedge C_2 \wedge \ldots \wedge C_n$;
(2) OR: $C_1 \vee C_2 \vee \ldots \vee C_m$;
(3) *n* OUT of *m*: $\{n, \quad C_1 \vee C_2 \vee \ldots \vee C_m\}$;
More generally, complex type:
(4) $G_1 \wedge G_2 \wedge \ldots \wedge G_l$,
G_i (i = 1, 2, …, m) is a condition expression of '*n* OUT OF

m' type.

For example, one of the most commonly-used prerequisite conditions is "*to take course C, learners must take X course out of Y courses, some are mandatory*". The condition represents that a student must take X prerequisite courses of course C, from Y courses to enable the course transition P. some of them are mandatory, Then the *pre*(*t*) can be described as follows:

(1) $\forall p$ in P, if the course transition which uses *p* as a *Done* place is a necessary course (the inscription of Arc connecting the input place and the transition is *MAND-ATORY*) then $M(p) > 0$; and

(2) $(\sum_{p} M(p) \times I_{-}(p, t)) \geq X$.

IV. Petri Net Markup Language

Petri Net Markup Language (PNML) is an XML based standard format for the interchange of Petri nets. The main elements in the PNML file for a curriculum are Transition, Place, and Arc. These three elements represent the three kinds of basic elements in a Curriculum Petri Net. Each of those elements has its sub-elements. To represent curriculum Petri nets using PNML, we do not have any element for the firing rules in the PNML, because we do not use a standard firing rule to fire the course Petri nets automatically. Instead, we set up a special business rule that implemented to fire the transitions of curriculum Petri nets.

In order to illustrate the PNML format, Figure 3 shows a very simple sub-net of a curriculum Petri net shown in Figure 1 and its corresponding PNML description. In between the <place id="p1"> and </place> tags, everything about that place is described. The value of its initial marking is contained between the tags of that name. The transitions are similarly described, whilst the arcs record their starts and ends as the nodes which they connect.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<pnml>
  <net id="n1" type="BlackTokenNet">
    <name> An Example of Curriculum Petri Net </name>
      <place id="READYCOMP501">
          <marking> <value>1</value> </marking>
           <name>
                  <value>READYCOMP501</value>
           </name>

<initialMarking><value>1</value></initialMarking>
    </place>
    … …
   <arc id="a1" source="READYCOMP501"

target="COMP501">
       <inscription>

<requirement>MANDATORY</requirement>
                <direction>input</direction>
                <value>0</value>
       </inscription>
          </arc>
    … …
    <transition id="COMP501">
        <name>
              <value>1</value>
        </name>
    </transition>
 </net>
</pnml>
```

Figure 3: An example of PNML-represented curriculum Petri nets.

To represent the complex relation of prerequisite courses in the <value> in transition node of the PNML, we use a special expression

<value>
m_1, course1, course 2, …, course n_1;
m_2, course1, course 2, …, course n_2;
…
m_l, course 1, course 2, …, course n_l
<value>

($l > 0$, $m_i < n_i$, $i = 1, 2, \ldots, l$). Here symbol ';' separates course groups. l is the number of the groups. m_i is the number of the required courses and n_i is the number of the courses to be selected.

For example, the value element in the transition node of the PNML for COMP696 is:

```
<value>
        4,COMP501, COMP503, COMP504, COMP601,;
        5,COMP603,COMP604,COMP605,COMP607,
           COMP610, COMP695,COMP617,;
          </value>
```

Here symbol ';' separates course groups. So, COMP696 has two course groups as its prerequisite courses. In the first course group, there are four courses, and four courses are required to be taken in that course group. It actually means every course is MANDATORY. In the second course group, there are seven courses, and five courses are required to be taken in that course group.

In this value string, we can't tell which course is MANDATORY or OPTIONAL. We can get that information from the inscription of Arc connecting the input place and the transition. In this example, among all these prerequisite courses in the second course group, only COMP695 is MANDATORY.

V. Web-Based Program Knowledge Editor

We developed a Petri nets based knowledge editor by using an extended PNK [15] which is a Java-based Petri Net

Kernel (PNK) providing an infrastructure for bringing ideas for analyzing, for simulating, or for verifying Petri Nets into action. Petri nets can be loaded and saved in PNML [16], a language for the description of Petri nets based on XML. This Web-based knowledge editor provides administrators with a convenient tool for the construction and maintenance of program models by adding, updating, or deleting courses, prerequisites, co-requisites, and rules. Figure 4 and 5 show the four steps to adding a course into course dependency database using the developed Web-based program knowledge editor.

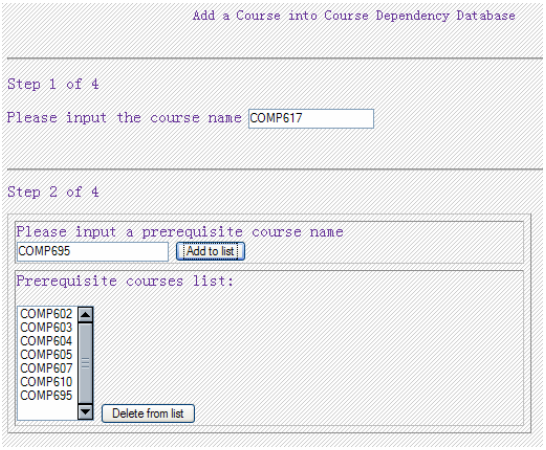

Figure 4: Step 1 and step 2 of adding a course into course dependency database using the Web-based program knowledge editor.

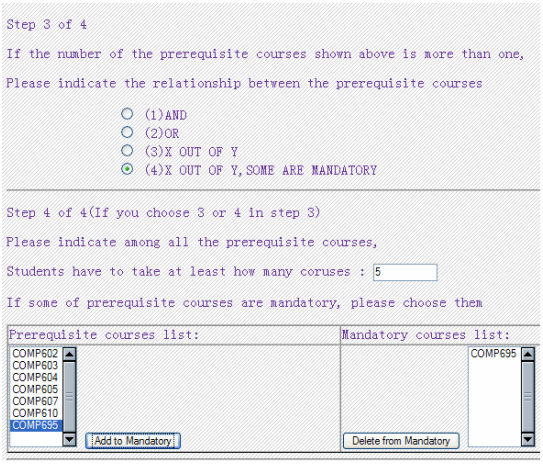

Figure 5: Step 3 and step 4 of adding a course into course dependency using Web-based program knowledge editor.

We applied the ontology-driven software development methodology for Web services and agents for the Semantic Web to develop a program planning agent --- e-Advisor [17]. Maximizing openness and choice can lead to learners making poor choices, thus flexible systems need to be accompanied by strong advice, monitoring and support systems [18]. Our continuing challenge is to maximize the quality and, availability of this support while continually reducing its cost. Given the wide number of courses choices or paths through the curriculum, it is important that each student be assigned an academic advisor who is a faculty member of the School of Computing and Information Systems. To be effective, advisors and faculty need to understand each individual student's educational and career objectives, their time and financial constraints, and their progress through the program. These can then be matched against degree requirements, course start dates, course availability, and prerequisites in order to generate a personalized course path. The provision of effective program advice for the average of 30 graduate students per faculty member is a time consuming task. In order to meet the students' needs and to reduce the workload of the advisors, we are developing an intelligent agent able to provide the students with an automated MSc IS Graduate program scheduling service (GPSS). On behalf of the students' advisors, the agent will assist them in generating *up-to-date*, *personalized*, and *optimal* study plans by constantly monitoring and utilizing related information resources.

The plan generation is executed by the Petri net program in the MSc IS ontology as a constraint-solving process. To communicate with Web services, the scheduling agent has to wrap its messages as SOAP messages. We are using Axis (http://ws.apache.org/axis/) and Java to build and deploy all Web services.

VI. Conclusions

The curriculum P/T net model proposed in this paper is able to represent accurately and dynamically the course-taking procedure for a given curriculum, provide a graphic tool for knowledge representation of the type of precedence relations constraints, provide a powerful simulation tool for program planning, give all possible solutions/study plans by simulation tracing or reachability analysis.

We are working on the verification (model-checking) module of curriculum Petri nets based program models and Web Services for Accessing PNML-represented program models. We are also exploring how to apply this approach to modeling concept ontology for developing intelligent e-learning systems.

Acknowledgement

We would like to thank National Science & Engineering Research Council (NSERC) of Canada and Athabasca University for sponsoring this research project.

References

[1] Golumbic, M. C., M. Markovich, S. Tsur, and U. J. Schild, A Knowledge-Based Expert System for Student Advising, *IEEE Transactions on Education*, Vol. E-29, No. 2, May 1986，pp 120-124。

[2] Cutright, K. W., R. Williams and D. Debald, Design of a PC-based Expert System for Academic Advising, *Computers & Industrial*

Engineering, Vol. 21, No. 1-4, pp.423-427, 1991.
[3] Gunadhi, H., K-H Lim, W-Y Yeong, PACE: a planning advisor on curriculum and enrolment, *Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS'95),* 01 04 - 01, 1995, Hawaii, USA, pp.23-31.
[4] Mallory, J. R., Using Current Artificial Intelligence Techniques to Advise Students, *Computer Applications in Engineering Education*, Vol. 1 (2), pp. 173-177, 1993.
[5] Kim, J. J., A Framework of an Integrated Knowledge-Based System for Academic Advising, *Int. J. Engng. Ed.* Vol. 8, No. 6, pp. 427-435, 1992
[6] Napper, S. A., K. V. Bertrand and B. L. Robey, REGAD: An Expert System for Registration Advising, *Int. J. Engng. Ed*. Vol. 8, No. 4, pp. 258-263, 1992.
[7] Miller, S. L., and L. G. Occeña, Design and Development of an Expert System for Undergraduate Course Advising, *Int. J. Engng. Ed.* Vol. 8, No. 1, pp. 43-55, 1992.
[8] Reisig, W., EATCS, Monographs on Theoretical Computer Science, W.Brauer, G. Rozenberg, A. Salomaa (Eds.), Springer Verlag, Berlin, 1985.
[9] Marinescu, D. C., *Internet-Based Workflow Management: Towards a Semantic Web*, John Wiley & Sons, ISBN 0-471-43962-2
[10] Nazareth, D. L., Investigating the Applicability of Petri nets for Rule-based system verification, *IEEE Trans. on Knowledge and Data Engineering*, vol.4, no.3, 1993, pp.402-415.
[11]Portinale, L., Behavioral Petri Nets: A Model for Diagnostic Knowledge Representation and Reasoning, *IEEE Trans. on SMC –Parts B*, vol. 27, no. 2, 1997, pp184-195
[12] Perdu, Didier M. and Alexander H. Levis, Requirements Determination using the Cube Tool Methodology and Petri Nets, *IEEE Trans. on SMC*, vol.23, no.5, 1993, pp1255-1264
[13] Muro-Medrano, P. R., J. A. Banares, and J. Luis Villarroel, Knowledge Representation-Oriented Nets for Discrete Event System Applications, *IEEE Trans. on SMC --- Part A: Systems and Humans*, vol. 28, No.2, March 1998 pp.183-198
[14] Lee, K., and M. Y. Jung, Petri Net Application in Flexible Process Planning, *Computers and Industrial Engineering*, Vol. 27, No. 1-4, pp.505-508, 1994
[15] PNK: http://www.informatik.hu-berlin.de/top/pnk/.
[16] Billington, J., S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, M. Weber, The Petri Net Markup Language: Concepts, Technology, and Tools, *ICATPN 2003*, Eindhoven, Netherlands, June 2003
[17] Lin, F, P. Holt, S. Leung, Q. Lin, A Multi-Agent and Service-Oriented Architecture for Developing Adaptive E-Learning Systems, *IJCEELL Special Issue on Semantic Web in E-Learning* (to appear）
[18] Tait, A., & Mills, R. (2003). *Rethinking learner support in distance education: Change and continuity in an international context*. London: Routledge.