INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 9(4):397-407, August, 2014.

Petri Net Based Modelling of a Career Syllabus

R. Carvajal-Schiaffino, L. Firinguetti-Limone

Rubén Carvajal-Schiaffino*[1]
Departamento de Matemática y Ciencia de la Computación
Universidad de Santiago de Chile
Av. B. O'Higgins 3363, Santiago de Chile
*Corresponding author: ruben.carvajal@usach.cl

Luis Firinguetti-Limone[2]
Departamento de Estadística
Universidad del Bío-Bío
Av. Collao 1202 - Concepción, Chile
lfiringu@ubiobio.cl

Abstract: A syllabus is a set of courses, their prerequisites and a set of rules defining the continuance of a student in an academic program. The manual verification of the possible contradictions between prerequisites and the set of rules is a difficult task due to the large number of possible cases. In this article we present the different stages in the verification of a syllabus, its modelling and analysis using Petri nets, and suggested ways in which this may be used by the university administration in the decision making process.
Keywords: Decision support systems, system modelling, system verification, educational planning, Petri nets.

1 Introduction

In many countries student retention rates are under scrutiny as part of government funding policy of higher education institutions. As such, retention rates are an increasingly important issue for these institutions ([2]).

It has been found that structural characteristics of higher education institutions, such as enrollment size, selectivity and control, have significant associations with student drop-out (see [5, 15, 16]). In this regard the consistency of a career syllabus may well play a significant role in student persistence/drop-out.

We may define a syllabus as a set of courses, their sequence, the institutional rules defining the continuance of a student in an academic program (number of credits, number of times a course can be repeated, student progress schedule, etc.).

In particular, student progress schedules, which define minimum requirements for continuance in an academic program, are set up by many institutions of higher education (see for instance [13, 14, 17, 18])

Petri net modelling can be an important tool to support decision making at different levels and types of organizations (see for example [4, 6, 10]). In this article we propose a Petri net model of a syllabus, to help the decision making process of the university administrators, students and the course lecturers. The administration may use this model to verify the internal consistency of

[1]Partially funded by DICYT-USACH under grant CS 060933
[2]Partially funded by the Universidad del Bío-Bío under grant DIUBB-083308 3/R

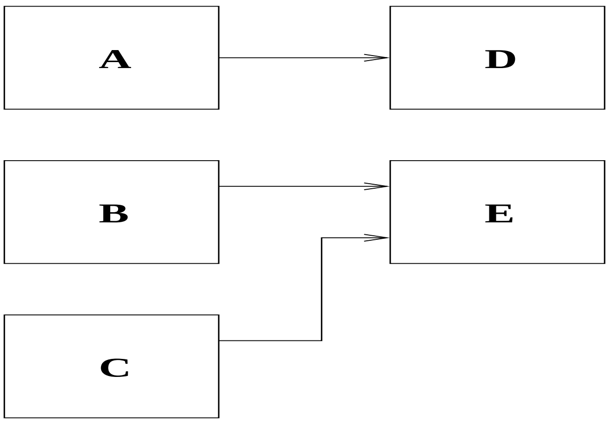

Figure 1: A prerequisites graph

an existing and/or under development syllabus; to plan the student career progress; to establish the risk level of a student drop-out.

In Section 2 we discuss the main characteristics of Petri nets, its feasibility to represent the prerequisites and the analysis of the resulting model. In Section 3 we demonstrate trough an example the stages in the modelling of a rule added to a Petri net representing the prerequisites; an analysis is made of the model obtained and we discuss some related work. Finally, in Section 4 we conclude, presenting our findings and suggestions for future work.

2 Modelling Syllabus Rules

In this article we consider the following rules in a career syllabus:

1.- Prerequisites between courses.

2.- Maximum number of times a course can be repeated.

3.- Maximum duration of stay in the academic program.

4.- Minimum number of courses to be taken during each academic period.

A Petri net [7,8] allows the formal mode description of systems whose dynamics is characterized by *Concurrency, Synchronization, Mutual exclusion, Conflicts*. Prerequisites present all these characteristics, except mutual exclusion. For instance, in Figure 1 we can find concurrency of courses *A, B, C* since, as already mentioned, they have to be taken at the beginning of the program. Another situation of concurrency may occur between courses *B, D* if a student passes courses *A, C* but fails course *B*.

Synchronization is present if to take a course it is necessary to pass more than one, as is the case of *E* since it is necessary to pass *B, C*.

A conflict does not appear explicitly in a prerequisite graph, nevertheless it is present since a course may or may not be passed.

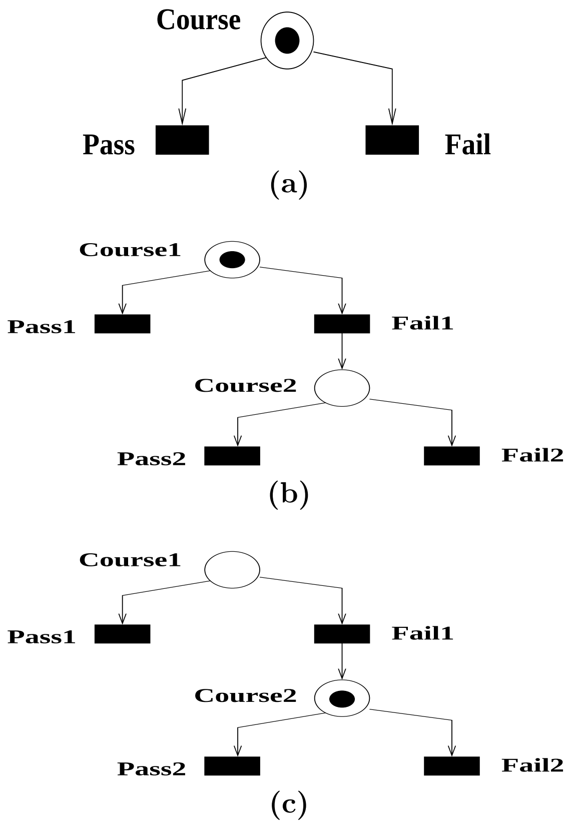

Figure 2: A course

2.1 Modelling Prerequisites and Maximum Number of Failures

Formally, the prerequisites are represented as an acyclic directed graph where each node represents a course and the arcs represent the dependencies (prerequisites) between courses. For example, Figure 1 may represent five courses *A, B, C, D, E* where the first three are the courses to be taken at the beginning of the academic program. The arc joining *A* and *D* pinpoints that to take course *D* it is necessary to pass course *A* and the arcs joining *B, E* and *C, E* indicate that to take course *E* it is necessary to pass courses *B* and *C*.

In Petri nets terms a course is represented by a place, whereas a place with a token indicates that the course is being taken. The activities that can be performed while taking a course are pass or fail. This is represented in Figure 2a.

Since a course can be taken at most twice, this may be represented as in Figure 2b. A token in place Course1 pinpoints that a student is taking a course for the first time. Firing the transition Fail1 indicates that the course was failed, having to be taken again (this is represented by a token in Course2). Firing the transition Fail2 indicates that the student failed Course a second time (Figure 2c), causing its elimination from the program. On the other hand firing transitions Pass1 and Pass2 pinpoint that Course was passed in the first or second instance respectively.

The Petri net representing a complete prerequisites graph is shown in Figure 3. Place *Enrolled* denote the fact that a student is registered in the academic program. Places *A1, A2, B1,*

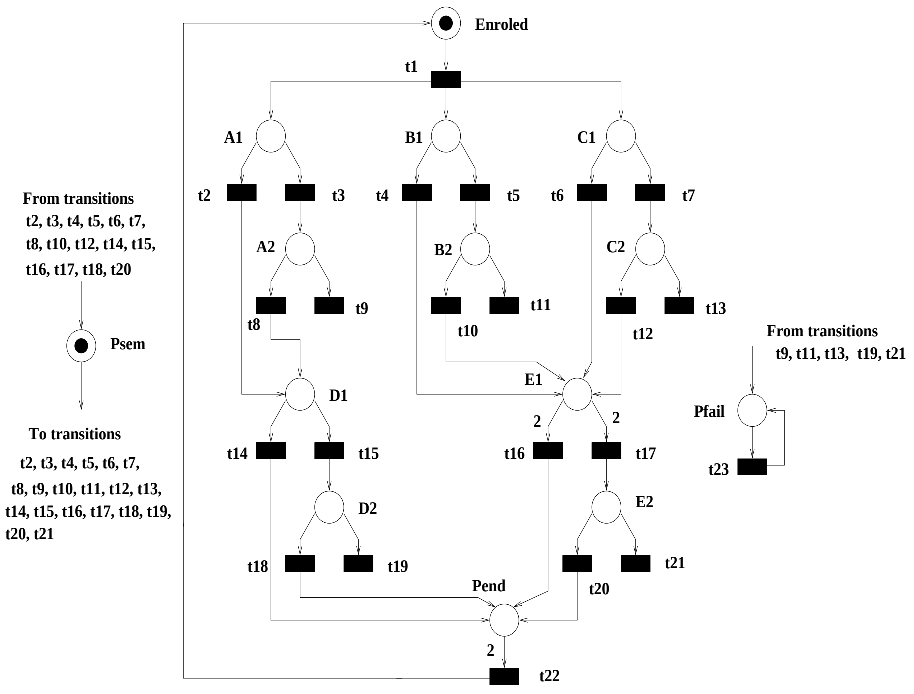
Enroled
t1
A1
B1
C1
t2
t3
t4
t5
t6
t7
A2
B2
C2
t8
t9
t10
t11
t12
t13
D1
E1
t14
t15
t16
t17
D2
E2
t18
t19
t20
t21
Pend
t22
Pfail
t23
Psem
From transitions t2, t3, t4, t5, t6, t7, t8, t10, t12, t14, t15, t16, t17, t18, t20
To transitions t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21
From transitions t9, t11, t13, t19, t21
2
2
2

Figure 3: A net representing the prerequisites graph of figure 1

B2, C1, C2, D1, D2, E1, E2 represent different possibilities of passing or failing the courses. The place *Pend* represents a student having taken all the courses of the program. Transition *t1* indicates the beginning of an academic period and firing it puts tokens in places *A1, B1* and *C1.* This global state indicates that a student, once enrolled, must take courses A, B, C. Firing the transitions *t2, t4, t6, t14, t16* respectively represent passing courses A, B, C, D and E the first time these courses are taken. On the other hand, firing the transitions *t8, t10, t12, t18, t20* represents passing the same courses but in the second opportunity. First time failures are produced when transitions *t3, t5, t7, t15, t17* are fired. Second time failures, and elimination from the program, are produced when transitions *t9, t11, t13, t19, t21* are fired.

Since failing a course twice results in the elimination of a student from the program, it is necessary to exclude the possibility of a student taking courses. For this we use place *Psem*, enabling transitions *t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20 ,t21.* Transitions *t2, t3, t4, t5, t6, t7, t8, t10, t12, t14, t16, t18, t20* return a token to *Psem*, but firing transitions *t9, t11, t13, t19, t21* represent failing a course a second time, not allowing any further courses to be taken. These failure transitions lead to the state *Pfail* from which it is not possible to escape.

To verify the properties of boundedness and liveness the transition *t22* is fired, putting a token in place *Enroled.*

2.2 Counting the Number of Periods to Conclude a Program

The model discussed on the previous Section allows the direct verification of the conditions of elimination from the program by failing a course a second time. This model satisfies three important properties: boundedness, liveness and absence of deadlock, which are verified by computing P-invariants, T-invariants and generating the reachability graph, respectively.

```
Algorithm GenGraph
Input: G, a prerequisite graph;
Output: H, a graph of valid courses taken in each period;
1.- Let V, vertex set of H
2.- Add vertex labelled $c_1 c_2 \ldots c_k$ to V
3.- Q $\leftarrow$ EnQueue(V)
4.- While (Q $\neq$ Empty) do
5.-     x $\leftarrow$ Dequeue(Q);
6.-     Let Y be all valid assignments from x taking
        into account the requisites of G
7.-     For All $y_i \in Y$ add an arc between x and $y_i$
8.-     EnQueue($y_i$)
9.- EndWhile
```

Figure 4: The algorithm.

We have carried the automated analysis of the Petri net model with the tool INA [9].

The number of academic periods necessary to complete the program is taken as a prerequisite for two reasons: to limit the maximum duration of stay in the program and to establish a minimum number of courses to be taken each academic period. Typically, the maximum duration of stay is 1.5 times the duration of the academic program. In our example the duration of the program would be 2 periods and the maximum duration of stay would be 3 periods.

A question to be answered is how to compute the number of academic periods (semesters, years, etc.) required to complete the academic program. To answer this question we must take into account that there are many performance cases. These vary from the optimum case: not failing any course, to the worst case: failing each course once. The model reachability graph contains all these cases, but we must consider that each state represents the firing of a transition not allowing the direct representation of the simultaneous passing/failing of several courses. However relevant information to count the number of periods is given only by states 1, 2, 5 and 7. State 1 (*Enrolled* + *Psem*) represents a student enrolling in the program the first time; state 2 (*A1* + *B1* + *C1* + *Psem*) represents a student sitting for the first time and simultaneously courses *A, B, C*; state 5 (*D1* + *2E1* + *Psem*) represents a student having passed all courses in the first period and sitting now courses *D, E*. The two tokens in place *E1* represents the fact that course *E* has two prerequisites *B, C*. Finally, state *2Pend* + *Psem* represents a student having taken all program courses.

From the previous discussion it may be deduced that, although the reachability graph contains all the states of the model and gives all valid sequences to complete the program (without time limitations) an additional time effort is necessary to recover only the states representing an academic period. In our example the reachability graph contains 96 states, but only 17 of the are relevant for our analysis.

A Feasible Approach to Counting Academic Periods

This problem can be solved by computing the model t-semiflows because, conceptually, they represent the sequence of fired transitions (in our case passing or failing sequences) to comeback to the initial state. If we regard as the initial state the enrollment of a student in the program and the possibility of taking courses, then any t-invariant will be a sequence of passing/failing

T-Semiflow	Number of Failures	Failed Courses	Number of Periods
t1 + t2 + t4 + t6 + t14 + t16 + t22	0		2
t1 + t3 + t4 + t6 + t8 + t14 + t16 + t22	1	A	3
t1 + t2 + t5 + t6 + t10 + t14 + t16 + t22	1	B	3
t1 + t2 + t4 + t7 + t12 + t14 + t16 + t22	1	C	3
t1 + t2 + t4 + t6 + t15 + t16 + t18 + t22	1	D	3
t1 + t2 + t4 + t6 + t14 + t17 + t20 + t22	1	E	3
t1 + t2 + t5 + t7 + t10 + t12 + t14 + t16 + t22	2	B,C	3
t1 + t2 + t5 + t6 + t10 + t15 + t16 + t18 + t22	2	B,D	3
t1 + t2 + t5 + t6 + t10 + t14 + t17 + t20 + t22	2	B,E	4
t1 + t2 + t4 + t7 + t12 + t15 + t16 + t18 + t22	2	C,D	3
t1 + t2 + t4 + t7 + t12 + t14 + t17 + t20 + t22	2	C,E	4
t1 + t2 + t4 + t6 + t15 + t17 + t18 + t20 + t22	2	D,E	3
t1 + t3 + t5 + t6 + t8 + t10 + t14 + t16 + t22	2	A,B	3
t1 + t3 + t4 + t7 + t8 + t12 + t14 + t16 + t22	2	A,C	3
t1 + t3 + t4 + t6 + t8 + t15 + t16 + t18 + t22	2	A,D	4
t1 + t3 + t4 + t6 + t8 + t14 + t17 + t20 + t22	2	A,E	3
t1 + t2 + t5 + t7 + t10 + t12 + t15 + t16 + t18 + t22	3	B,C,D	3
t1 + t2 + t5 + t7 + t10 + t12 + t14 + t17 + t20 + t22	3	B,C,E	4
t1 + t2 + t5 + t6 + t10 + t15 + t17 + t18 + t20 + t22	3	B,D,E	4
t1 + t2 + t4 + t7 + t12 + t15 + t17 + t18 + t20 + t22	3	C,D,E	4
t1 + t3 + t5 + t7 + t8 + t10 + t12 + t14 + t16 + t22	3	A,B,C	3
t1 + t3 + t5 + t6 + t8 + t10 + t15 + t16 + t18 + t22	3	A,B,D	4
t1 + t3 + t5 + t6 + t8 + t10 + t14 + t17 + t20 + t22	3	A,B,E	4
t1 + t3 + t4 + t7 + t8 + t12 + t15 + t16 + t18 + t22	3	A,C,D	4
t1 + t3 + t4 + t7 + t8 + t12 + t14 + t17 + t20 + t22	3	A,C,E	4
t1 + t3 + t4 + t6 + t8 + t15 + t17 + t18 + t20 + t22	3	A,D,E	4
t1 + t2 + t5 + t7 + t10 + t12 + t15 + t17 + t18 + t20 + t22	4	B,C,D,E	4
t1 + t3 + t5 + t7 + t8 + t10 + t12 + t15 + t16 + t18 + t22	4	A,B,C,D	4
t1 + t3 + t5 + t7 + t8 + t10 + t12 + t14 + t17 + t20 + t22	4	A,B,C,E	4
t1 + t3 + t5 + t6 + t8 + t10 + t15 + t17 + t18 + t20 + t22	4	A,B,D,E	4
t1 + t3 + t4 + t7 + t8 + t12 + t15 + t17 + t18 + t20 + t22	4	A,C,D,E	4
t1 + t3 + t5 + t7 + t8 + t10 + t12 + t15 + t17 + t18 + t20 + t22	5	A,B,C,D,E	4

Table 1: T-semiflows obtained from the net in figure 3

a course such that the last transition fired will be *t22* which allows to comeback to the initial state.

The net possess 33 t-semiflows (Table 1), but only 32 of them are valid for computing the number of academic periods. For instance, the firing sequence necessary to take all the courses in the program failing none is *t1 + t2 + t4 + t6 + t14 + t16 + t22*. With this information it is possible to determine the periods of time taken. For this it is necessary to take into account the places acting as synchronization points, in other words courses with more than one prerequisite. As shown in Table 1, from the total of 32 valid sequences to sit courses without failing more than twice, only 15 allow completion of the program without entering in conflict with the maximum time of continuance in the program. Note that there are situations where failing two courses (*B,E* or *C,E* or *A,D*) results in the elimination of the program.

However, in the case of programs of study with many courses, counting the t-semiflows of the corresponding Petri nets is not possible. The reason for this is that the number of t-semiflows is exponential with respect to the number of courses. For example, for a program of study with 16 courses (4 academic periods), the corresponding reachability graph contains 47,935 states and 65,536 valid sequences for sitting the courses failing at most once some of them.

To solve the problem of the maximum number of semiflows that can be computed with a tool to analyze Petri nets, we propose an algorithm (see Fig. 4) that allows the generation of valid sequences to fulfill a program with no more than two failures in each course. This algorithm generates a directed graph in which each vertex contains a set of courses that can be taken in each academic period.The initial vertex of this graph contains the set of courses to be taken at the beginning of the program. Then, -taking into account the number of resits and the restrictions asserted by the prerequisite graph-, from each vertex are generated the vertices containing the set of courses to be taken next.

Figure 5 shows the graph of the syllabus of 5 courses given in Figure 1; In this case the first vertex, $a_1b_1c_1$, represents courses *A,B,C* which are taken for the first time. The adjacent vertices

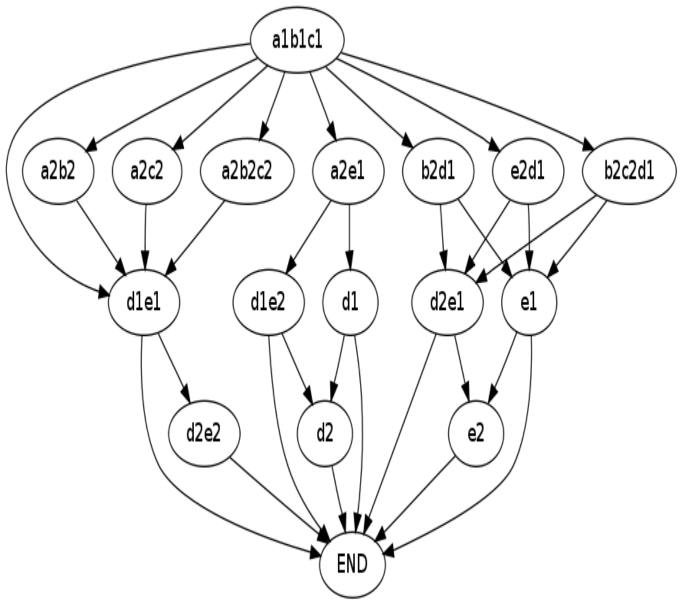

Figure 5: Graph generated with algorithm of figure 4 using the graph of figure 1 as input

represent the courses that are taken in the following period having into account the number of repetitions and restrictions of the prerequisite graph. As can be observed, in this case the number of vertices is 17. A valid sequence of courses taken (having into account repetitions) is obtained if one follows the graph from the initial vertex to vertex *END*. For instance the sequence $a_1b_1c_1, d_1e_1, END$ represents passing all courses, failing none. In the opposite case, the sequence $a_1b_1c_1, a_2b_2c_2, d_1e_1, d_2e_2, END$ represents taking all courses twice. The number of periods taken is given by the number of vertices visited minus 1.

The number of trajectories of all paths in the graph coincides with the number of t-semiflows of the Petri net representing the prerequisites graph with the advantage of greater velocity and permitting the analysis of programs with a larger number of courses.

2.3 Minimum Number of Courses to Be taken During Each Academic Period

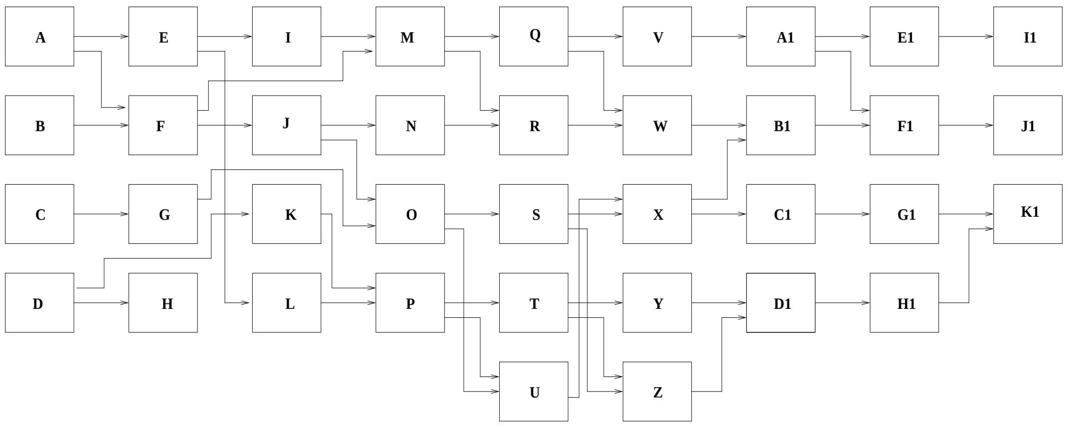

Figure 6: A prerequisites graph for a nine periods program

Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
	2	3	3	3	1

Table 2: Minimum number of courses passed for the academic program

Number of courses taken	Frequency
1	8
2	30
3	72
4	107
5	79
6	24

Table 3: Frequency of the number of courses taken for the academic program

To illustrate the analysis of this rule we use the algorithm of Figure 4, with the first four academic periods of the prerequisite graph of Figure 6. The reduced graph of this model generates 320 states and 65,536 valid sequences. The minimum number of courses passed per semester is given in Table 2.

The number of valid courses taken each semester varies from 1 to 6. Table 3 shows the distribution of the 320 valid sequences. Except for the first, second and sixth period, the minimum number of courses passed per semester is 3. From this, two situations must be analyzed:

a) less than 3 courses are taken

b) more than 4 courses are taken

If less than 3 courses are taken and these are from the last part of the syllabus, no problem will arise, since the student will be taking the courses necessary to complete the program. This will not be the case if the courses are taken before the final part of the program.

For a course taken there are 8 possible cases, discarding the 4 cases that represent taking only courses *M, N, O, P* a second time, there remain four cases to analyze, corresponding to taking courses *F, I, J* a second time or else *P* for the first time.

Path
$A_1B_1C_1D_1 \rightarrow B_2C_2D_2F_1 \rightarrow F_1G_1H_1K_1L_1 \rightarrow F_2$
$A_1B_1C_1D_1 \rightarrow E_1F_1G_1H_1K_1 \rightarrow E_2G_2J_1 \rightarrow I_1L_1N_1O_1 \rightarrow I_2$
$A_1B_1C_1D_1 \rightarrow B_2C_2D_2E_1 \rightarrow F_1G_1H_1I_1K_1L_1 \rightarrow G_2J_1M_1 \rightarrow J_2$
$A_1B_1C_1D_1 \rightarrow C_2D_2E_1F_1 \rightarrow E_2G_1H_1J_1K_1 \rightarrow H_2I_1K_2L_1N_1O_1 \rightarrow L_2M_1N_2 \rightarrow P_1$

Table 4: Sequences of courses taken enabling to take only one course

When it is feasible to take more than 4 courses in each academic period, there are 79 situations with 5 courses (Table 5) and 24 with 6 courses (Table 6). The analysis of these cases are useful to limit the number of courses a student can take, since there is a larger number of situations where a student takes courses for a second time. The proposed method allows the analysis of the student progress schedule (that is to say the total number of courses passed up to a certain period) with the valid sequences generated. This analysis allows to detect if failing a course, which is prerequisite of others, produces the elimination of a student because it may not be possible to take anymore courses to comply with the student progress schedule.

Courses taken for the first time	Total
5	3
4	12
3	19
2	21
1	14
0	10

Table 5: Distribution of courses taken for the first time in groups of five

Courses Taken for the first time	Total
6	2
5	3
4	4
3	5
2	5
1	3
0	2

Table 6: Distribution of courses taken for the first time in groups of six

2.4 Decision Support

Of common interest to university administrators, students and course lecturers is to reduce student drop-out. Our model may help to plan the courses taken by students to achieve this objective.

Valid sequences provided by the previously developed algorithm would help academic administrators, students and lecturers in making decisions. In particular, administrators may use this information to detect which students are at risk of dropping-out and suggest which courses are the most convenient to achieve the objective of completing the academic program.

On the other hand lecturers, once detected students in risks of drop-out, may advice students and administrators to take some corrective actions to reduce this risk.

2.5 Practical Evaluation

We now evaluate the algorithm used to generate the valid sequences of the possible courses to take with no more than two repetitions. Using as entry the graph of prerequisites of Figure 6 we have run the algorithm incrementing the number of levels from 4 to 9.

The results obtained are shown in Figure 7. It can be observed that for the complete syllabus there are 3,838 valid sequences with 93,244,504,769 possible paths. Processing time was almost an hour. The experiments were made with a computer with a 2.8 GHz INTEL processor and 4 Gb RAM.

Another experiment made consisted in counting the number of sequences leading to a situation were the number of courses to take is insufficient, (in our case less than 3). We found 70,207,548,816 cases. We also counted the number of situations were the maximum number of semesters was exceeded. We found 18,475,875,736 of these cases.

Levels	Vertices	Paths	Time
4	464	65,537	0.002s
5	1,184	2,097,153	0.126s
6	2,182	67,108,865	2.497s
7	2,862	882,837,761	29.049s
8	3,440	12,598,171,649	433.505s
9	3,838	93,244,504,769	3,463.607s

Figure 7: Experimental evaluation

3 Results and Discussion

The work presented in [3] uses Petri nets to assist curricula development, modelling the sequence of courses to be taken by students in an academic program. [11] describes the application of model checking to the automatic planning and synthesis of student careers under a set of requirements and to the automatic verification of the coherence of syllabi subject to a set of rules. For this it is suggested to provide the students with a sort of electronic advisor of studies. However, it is recognized that the verification of the coherence of syllabi with the set of rules is by far the most difficult problem and out of the reach of standard technology of information systems, because it requires a (double) exhaustive search engine. The basic step is to explore all possible course combinations to find a feasible set of courses which satisfies the given set of rules and prerequisites to qualify for the university degree chosen by the student. The authors propose to encode and solve the problems of the of the synthesis of student careers and of the verification of the syllabi with respect to the set of rules as computational tree logic model checking problems.

4 Conclusions

We have presented a methodology for the analysis and verification of prerequisites of courses of a syllabus modelled with a Petri net. Also, we have verified that the model is consistent and capable of being generated in automatic mode.

Besides, we have indicated how this model may be used by administrators and lectures to suggest students corrective actions to reduce drop-out caused by a bad selection of course sequences.

The next stage in this research is to enrich the model with the representation of the number credits given to each course. Also an advance table will be included which consists in verifying whether student complies with the minimum number of credits required since he or she enrolled in the program. In a subsequent stage we plan to apply the model to a group of students and produce a quantitative analysis using the stochastic extensions of Petri nets [1].

Bibliography

[1] Ajmone Marsan M., Balbo G., Conte G., Donatelli S., and Franceschinis G. (1995); Modelling with Generalized Stochastic Petri Nets, *Series in Parallel Computing*, John Wiley & Sons.

[2] Doherty W (2006); An Analysis of Multiple Factors Affecting Retention in Web-Based Community College Courses, *The Internet and Higher Education*, 9:245-255.

[3] Ferraris M., Midoro V. and Olimpo G. (1984); Petri Nets as a Modelling Tool in the Development of CAL Software, *Computers and Education*, 8:41-49.

[4] Jahangirian M., Eldabi T., Naseer A., Stergioulas L., Young T. (2010); Simulation in Manufacturing and Business: A Review, *European Journal of Operational Research*, 203:1-13.

[5] Kim D. B.; The Effect of Loans on Students Degree Attainment: Differences by Student and Institutional Characteristics, *Harvard Educational Review*, 77(1):64-100.

[6] Rajabi B. A. and Lee S. P. (2009); Change Management in Business Process Modeling Based on Object Oriented Petri Net, *International Journal of Human and Social Sciences*, 4:13.

[7] Reisig W (1986); Petri Nets. An introduction, *EATCS Monographs on Theoretical Computer Science*, Springer.

[8] Reisig W. and Rozenberg G., editors (1998); Lectures on Petri Nets I: Basic Models, *Advances in Petri Nets*, LNCS 1491. Springer.

[9] Roch S. and Starke P. H. (1999); INA: Integrate Net Analizer, *Humboldt-Universität zu Berlin.*

[10] Salimifard K., Wright M. (2001); Petri Net-Based Modelling of Workflow Systems: An Overview, *European Journal of Operational Research*, 134(3):664-676.

[11] Sebastiani R., Tomasi A. and Giunchiglia F. (2001); Model Checking Syllabi and Student Careers, *Lecture Notes in Computer Science 2031.* Springer.

[12] Silva M., Teruel E., and Colom J. M (1998); Linear Algebraic and Linear Programming Techniques for Analysis of Place/Transition Net Systems. *In Reisig and Rozenberg [8]*, 308-309.

[13] Stevenson University (May 2011); Policies for Continuance and Progression in the Major, www.stevenson.edu/academics/nursing/policies.asp

[14] The College of William and Mary (May 2011); Continuance Requirements for Fulltime Students. www.wm.edu/offices/deanofstudents/policies/academic/contfulltime/index.php

[15] Titus M. A. (2004); An Examination of the Influence of Institutional Context on Student Persistence at 4-year Colleges and Universities: A Multilevel Approach, *Research in Higher Education*, 45(7):673-699.

[16] Titus M. A (2006); Understanding College Degree Completion of Students with Low Socioeconomic Status: The Influence of the Institutional Financial Context, *Research in Higher Education*, 47(4):371-398.

[17] Universitat Pompeu Fabra (May 2011); Regulation Governing the Progression System and the Continuance Rules for Undergraduate Courses, www.upf.edu/universitat/en/normativa/upf/normativa/grau/RD1393/permanencia/

[18] Wabash College. Continuance in College (May 2011); www.wabash.edu/academics/bulletin.cfm?site_code_id=967.