Sensors and Materials, Vol. 34, No. 6 (2022) 2229–2241
MYU Tokyo

S & M 2968

Basis for Deep Learning Model of Discrete Event System for Information Technology Course Design

Haidong Fu,[1,2] Yueguang Xie,[1] and Jih-Fu Tu[3*]

[1]School of Information Science and Technology, Northeast Normal University
No. 2555 Jingyue Ave., Changchun City, Jilin Province 130117, China
[2]Academic Affairs Office, Changchun University
No. 6543, Weixing Rd., Chaoyang Dist., Changchun City, Jilin 130022, China
[3]Department of Electrical Engineering, Lunghwa University of Science and Technology
No. 300, Sec. 1, Wanshou Rd., Guishan District, Taoyuan City 333326, Taiwan

(Received December 30, 2021; accepted April 6, 2022; online published April 21, 2022)

Keywords: core literacy, metaphysical, discrete event system, information technology teachers, teaching ability

The development of technologies is changing the traditional concept of teaching and learning in classrooms. In particular, information technology (IT) courses are rapidly adopting such changes as IT education's purpose is to cultivate students' capability to develop technologies by experiencing them in classrooms. In IT courses, deep learning has been used to improve teaching and learning ability through the use of sensing technology and related hardware/software. In teaching and learning, asynchronous, concurrent, and stochastic events occur because teachers and students have their protocols and behaviors in the related activities. Therefore, the design and teaching of courses should be based on a discrete event system that deals with discrete events. On the basis of this concept, we propose a deep learning model with the Petri net method to establish a logical modular system for IT course design and teaching. Because Petri nets are useful in simulation and analysis for the system modeling of asynchronous, concurrent, and stochastic events, we investigate how to use a Petri net to establish a deep learning model to develop and evaluate the curriculum of IT courses. The results of this study will contribute to building an efficient IT learning system that comprises sensing technology, information transmission, information processing, and feedback, which will require collaboration with pedagogical experts in the future.

1. Introduction

Some academic courses in information technology (IT) require teaching discrete event systems (DESs), in which Petri net methods are most widely used.[1,2] Recently, many studies on teaching logic, problem-solving ability, and the symbolization of natural languages have become important in the development of technologies used in artificial intelligence and natural language processing.[3] Morou and Kalospyros emphasized the importance of mathematical reasoning to improve the cognition of logic in teaching and learning.[4] Butchart and Handfield[5] argued the importance of philosophy, formal logic, and critical thinking even in peer instruction. James[6]

*Corresponding author: e-mail: tujihfu@gmail.com
https://doi.org/10.18494/SAM3826

ISSN 0914-4935

suggested adding logical components to improve teaching ability in IT courses.

A Petri net was first used by Engberg and Winskel[7] in the models of Girard's intuitionistic linear logic. It provided mathematical and graphical models to improve understanding of linear logic. The feasibility of Girard's linear logic was reviewed as a specific language for its interpretation in Petri nets.[8] Petri nets have also been used to teach algebraic calculus[9] and the relationship between logic and concurrency.[10]

Since they were proposed, Petri nets have been used in industrial and academic research.[11] Petri nets are often used for event-based simulation and analysis in manufacturing system modeling that demands the behavior of the system to be rendered in an event or activity. They are used for concurrent, distributed, nondeterministic, asynchronous, or stochastic events for system analysis and design description. Petri nets are also applied to analyze sensor behavior and diagnose sensors in a wireless sensor network.[12,13]

The graphical presentation of a Petri net[1,11,14] is similar to that of flow charts or networks used to convey visual information. A Petri net simultaneously describes the activities or events within a system and the changing status of the activities by using a state transition diagram.[14] Petri nets are regarded as a useful process definition tool in manufacturing process automation, traffic management, process control, and education.[5,15–19] Researchers used Petri nets to propose a flight control process model, message sequence charts, a performance analysis model,[18] a web service process, and pre-authentication in web service composition.[10,20] Timed Petri nets (TPNs) and fuzzy Petri nets are applied to a knowledge structure model for students' personalized optimal learning paths.[5] Kučera *et al*.[2] also used a Petri net to establish a model for learning and teaching microcontrollers.

On the basis of previous studies, we attempt to improve general teaching ability in IT courses by applying the Petri net to a deep learning model.[2] Nowadays, many sensing technologies are used in IT education because teaching IT with deep learning or machine learning may enable monitoring of students' learning activities and enable the exact and prompt evaluation of learning outcomes. For this, various types of hardware with sensing technologies are required for sensing drawings, writing, and algorithms created by students, and their attitude and posture. A deep learning model has been used to provide personalized learning for students in courses on statistics and AI. Applying deep learning methods to the learning experience requires sensing technology to save and deliver information about the learning process and outcomes. Many IT courses require students to create flow charts for logic and programs. To evaluate them, it is necessary to observe and monitor the learning process of students. The new Petri net model in this study provides the basis for developing a deep learning model to develop and effectively teach IT courses based on the data collected using sensing technologies.

2. Methods

2.1 Properties of Petri nets

Petri nets are modeling tools composed of places, transitions, tokens, and directed arcs, as shown in Table 1.[11,14,21,22] They have the following characteristics.

Table 1
Petri net basic compositional elements.

Elements	Symbols	Definition
Place	○	Represents usable resources or possible states occurring in system
Transition	▬	Represents place-changing event of system or beginning and ending of event
Token	●	Represents whether or not a system place exists and usable resource variables
Directed arc	⟶	Links place to transition, representing their input-output relationship

1) The dynamic behavior of an operating simulation system is expressed through the use of Petri net graphics.
2) In addition to graphical expression, Petri nets also provide formal semantics based on mathematics.
3) Petri nets enable many behavioral property analyses to verify a Petri net model.

The significance of individual places representing the state of procedures and resources of a system is expressed as circles. Transitions describing the activities and events of the system are expressed as line segments or rectangles. Places and transitions are connected by directed arcs to represent the causal relationship between the flow orders of place changes and events. Places contain at least one non-negative integer; these integers are indicated by black dots that are usually presented as tokens (●) as shown in Fig. 1.

Petri nets are expressed by using a combination of the four symbols and five elements (P, T, F, W, M_0). The elements are defined as follows.

(1) $P = \{ P_1, P_2, P_3, \ldots, P_m\}$ is a finite set of nodes.

(2) $T = \{t_1, t_2, t_3, \ldots, t_n\}$ is a finite set of transitions.

(3) $F \subseteq (P \times T) \cup (T \times P)$, where arc set F describes the link between the place and transition and represents the causal relationship between them, $(P \times T)$ represents a place-to-transition arc, and $(T \times P)$ represents a transition-to-place arc.

(4) $W : F \rightarrow \{1, 2, 3, \ldots\}$, where W is a set of weighting functions of tokens on the arc. We use $W(p, t)$ to express the weight of the place-to-transition arc and $W(t, p)$ to express the weight of the transition-to-place arc.

(5) $M_0 = \{M(P_1), M(P_2), \ldots, M(P_m)\} : P \rightarrow \{0, 1, 2, 3, \ldots\}$ is an initial token, where $M(P_1)$ represents the number of tokens within place P_1.

After constructing a model by using a Petri net, the basic characteristics of the model must be analyzed.

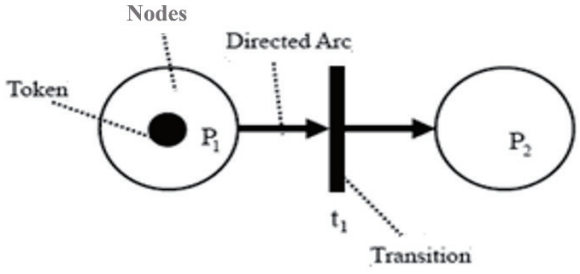


Fig. 1. Simple Petri net model.

The common characteristics of a Petri net model are listed as follows.[9,11,23]

(1) Liveness

If each saved transition within a model is kept in an orderly sequence, tokens from the initial distribution M_0 can reach any of the distributed situations within the model (M_1, M_2, …, M_n). The liveness of a model means that no complete deadlock occurs within a model. In other words, each activity occurs all the time.

(2) Boundedness

Under the distribution of any token within a model, the number of tokens within each place must not exceed the *K*-value limit, which is called "model boundedness". When $K = 1$, a model has "model safeness".

(3) Reachability

Reachability is a fundamental concept in the dynamic properties of a model to show that M_n is obtained after a token passes a series of transition firings. The ordered sequence of transition firings is expressed as $T = \{t_1, t_2, t_3, \ldots, t_n\}$.

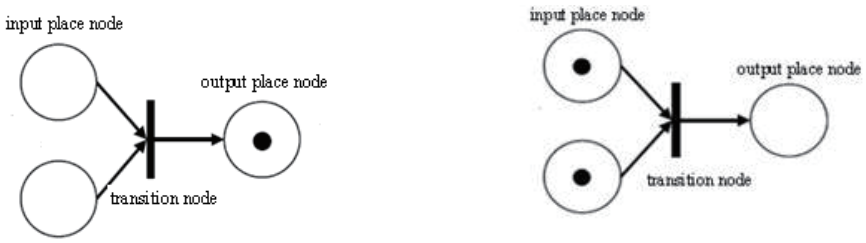
2.2 Dynamic behavior and analysis of Petri nets

In a Petri net model, not all input place nodes obtain tokens for a transition. A transition requires a condition to move. When there are several transitions with such conditions, various places are determined in a model. Regardless of whether firing occurs, a transition can be initiated randomly or according to a certain set of rules. Once the transition starts, the tokens of all input place nodes move to the output place nodes as shown in Fig. 2. At this point, the model proceeds to the next step and continues to infer the input of other places with other triggered transitions.[11,24]

A Petri net is a two-valued logical representation with fast deductive capability to handle parallel, real-time, scattering, and uncertain operations. The following is a list of the five basic Petri net modules used in this paper.

(1) **AND**-type module (**If** (P_1 *and* P_2 and … and P_n) **Then** (P_m))

As shown in Fig. 3, if all the input places P_1, …, P_n in transition *t* obtain tokens, transition *t* is triggered. When transition *t* occurs, all tokens connected to *t* are cleared. At the same time, a token is sent to output place P_m. At this point, the state of the Petri net module shifts to P_m at transition *t*. Then, the module is regarded as the ***AND*** computing operation of a logic gate.



(a) (b)

Fig. 2. Examples of firing and triggered nodes. (a) Transition. (b) Triggering transition.

(2) Event-conflict-type module

In the model shown in Fig. 4, P_1 is the only node with a non-negative integer. There are input place nodes for transitions t_1 and t_2 that are triggered, but only one node can move. This state is called the event conflict-type module, which requires that selections or decisions be made on the basis of t_1 and t_2. In other words, special evaluation functions are added to transitions t_1 and t_2. Then, we can use the provided information to select a transition node for its action.

(3) Concurrent-execution-type module

As shown in Fig. 5, there is only one token in node P_1. P_1 is the input node for transition t_1 that triggers transition t_2. If transition t_1 moves, the token at node P_1 passes to the nodes of the next levels of P_2, P_3, …, P_n, generating a concurrent event-execution phenomenon.

(4) Buffer-type module (**If** (P_n) **Then** (P_m))

As shown in Fig. 6, when input node P_n at transition t obtains a token, t is triggered. If transition t moves, the token at node P_n passes to the next level, P_m, which changes the status of the module from the original P_n to P_m.

(5) Looping-token-type module

As shown in Fig. 7, when node P_1 has only one token, the node is the same for the input at transition t_1 and the output at transition t_2. When t_1 is triggered, the token at P_1 passes to the next level P_2. The triggering of t_2 generates a transfer effect that creates a looping situation.

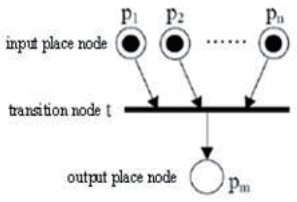


Fig. 3. **AND**-type module.

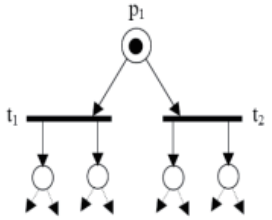


Fig. 4. Event-conflict-type module.

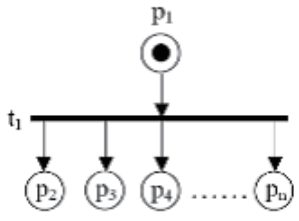


Fig. 5. Concurrent-execution-type module.

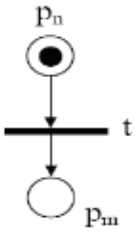


Fig. 6. Buffer-type module.

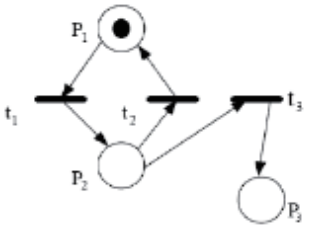


Fig. 7. Looping-token-type module.

2.3 Application of Petri net

Applying a Petri net to a development process only fulfills the basic requirement of the overall work. The purpose of the Petri net model is to understand the characteristics of a system using various analytical methods by detecting and verifying problems in the system. There are several methods of analysis for a Petri net.

(1) Reachability tree

A reachability tree establishes a treelike structure in which each node represents the current state when the moved place node at the previous level is sufficient to place the node at the next level. A reachability tree is used to analyze the boundedness and liveness of a Petri net. When the system is too large, the graphic becomes too complex to analyze. Thus, a reachability tree is appropriate for small systems. Figure 8 shows an example of a Petri net and its reachability tree. The initial marking state M_0 is (1, 0, 0, 0), which represents the availability of tokens at (P_1, P_2, P_3, P_4). The marking is set to 1 if token availability is confirmed. Otherwise, it is set to 0. The reachability tree allows a complete display of possible transition firing sequences and the marking distribution of the system.

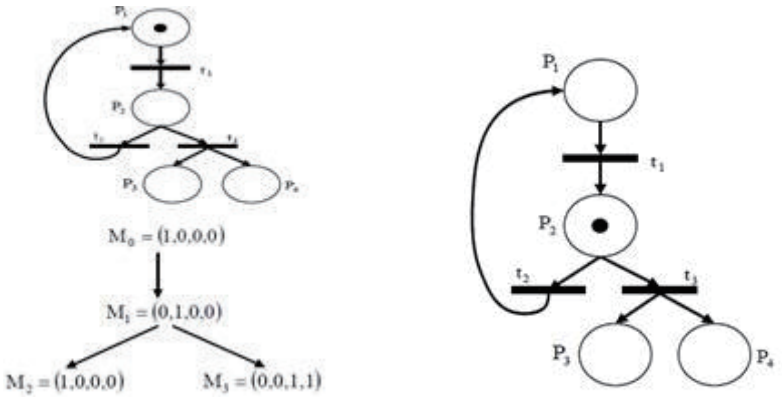
(2) Matrix equation

Equation (1) is used to infer the behavior of a Petri net. First, a system matrix is defined and used to indicate the flow relationship between places and transition *s*. The mathematical equation calculates the state after the transition.

The system matrix is derived from $a = a^+ - a^-$, where $an^+ = W(i, j)$ is the weight value from transition i to output place j and $a = W(i, j)$ is the weight value from input place j to transition i. The matrix equation is

$$M_i = M_{i-1} + a_n{}^* u_1, \tag{1}$$

where K = 1, 2, …, n, M_k pertains to the Kth state matrix with m × 1 columns, a_{ij} is the transition matrix of the system, and u_k indicates the Kth trigger matrix.[3,17] For the Petri net shown in Fig. 8(a), the corresponding system matrix is as follows.



(a) (b)

Fig. 8. Simple example of Petri net and its reachability tree. (a) Petri net. (b) Reachability tree

$$q_j = q_j^+ - q_j^- = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Here, a_{ij} represents the input matrix and a_{ij}^+ represents the output matrix of the system. Table 2 shows the matrices when t_2 moves. When t_2 is triggered, we obtain matrices with negative values according to Eq. (1). Because negative values in the matrices are unreasonable, the use of Eq. (2) is preferable.

2.3 Types of Petri nets

Events, conditional situations, processing synchronization, decision processing, and looping patterns can be described using Petri nets. Various Petri nets with different purposes exist, such as colored Petri nets, timed Petri nets, and hierarchy Petri nets.

(1) Colored Petri net (CPN)[20]

In a conventional Petri net, all tokens have the same value. In workflow applications, the tokens indicate different resources such as documents, programs, or participants. A CPN is thus used to mark the resources.

(2) Timed Petri net (TPN)[23]

To describe temporal aspects including performance evaluation and dynamic system scheduling, the concept of time must be introduced in the Petri net. Thus, a TPN was proposed for describing tasks involving time. A TPN has two different time delays: a residence time delay and a trigger time delay. In the residence time delay, each token must stop at each place for a certain time before it is triggered. A token for a stopping time that exceeds the delay time is defined as the liveness token. The liveness token is output to a transition. After a transition moves, its liveness token is sealed. The sealed token moves to another transition after a certain delay time. This is defined as the trigger time delay.

(3) Hierarchy Petri net (HPN)[10,16]

A Petri net for a large-scale process could produce complicated results. To obtain a simpler result, an HPN is required to render complex processes. It uses a hierarchical approach to reduce the complexity by having a modular workflow. The HPN reiterates the existing workflow to effectively avoid difficulties in the existing workflow.

Table 2
Adaption matrix of Petri net.

I	P_1	P_2	P_3	P_4	
t_1	1	0	0	0	a_{ij}^-
t_2	0	1	0	0	
t_3	0	1	0	0	
O	P_1	P_2	P_3	P_4	
t_1	0	1	0	0	a_{ij}^+
t_2	1	0	0	0	
t_3	0	0	1	1	

3. Results and Discussion

We use the modular construction of a Petri net for the operational flow of the development process with four basic components of the Petri net: places, transitions, tokens, and directed arcs. The development process is displayed using a Petri net diagram. The transition process in the Petri net includes the steps of notation, initial state, sequential operation, forking, joining, branch, selection, and loop.

A process diagram includes rectangular boxes for programs and rhomboidal boxes for decision-making procedures, which are shown as places. For an operating process to proceed to the next level, an action (a transition) is necessary. In a Petri net, the opening of a new case is defined as a token. A token is moved with the aim of completing the operating procedure at a certain level. Directed arcs represent the relationship between operating procedures and actions.

Figure 9 shows an ordinary workflow. Many programs are executed sequentially without branching or parallel behavior by following the operating sequence of working procedures A, B, and C. Figure 10 illustrates the sequential operation of the process flow using a Petri net.

Figure 11 presents Petri nets that display forking in the flow. When a token at P_1 moves at t_1, P_1 and P_2 generate their tokens at the same time to express their parallel behaviors in the logic program. Figures 11(a) and 11(b) show the decision-making branch forming a loop in the flow as expressed by the Petri net. Transition t_1 continuously moves until the condition for leaving the loop is satisfied; only then does t_1 stop firing. Figures 11(c) and 11(d) show the joining of the flow in the Petri net. Only P_1 and P_2 at t_1 show synchronous behavior in the logic programming. They also display the selections or event conflicts in the flow process. In this study, the operating procedure is set as shown in Fig. 11(d). They are further distinguished and discriminated as shown in Fig. 11(a). Figure 11(e) represents the conversion by the Petri net modules and the triggering of t_1 and t_2. However, only one transition can be triggered. This process is regarded as an event-conflict-type module, which must choose between t_1 and t_2.

The Petri net module used in this study is presented in Table 3. On the basis of the transition rules for the flow process of the Petri net, the previously adopted transition from a new development flow process to a Petri net module is shown in Fig. 12.

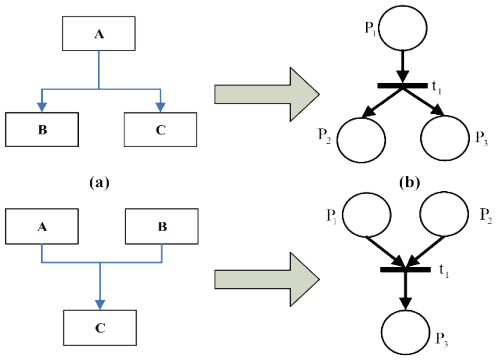


Fig. 9. (Color online) Flow of sequential operation.

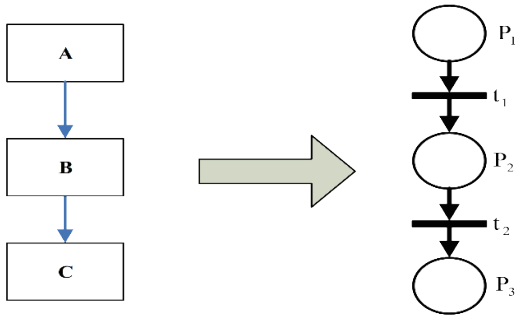


Fig. 10. (Color online) Flow of separation and import.

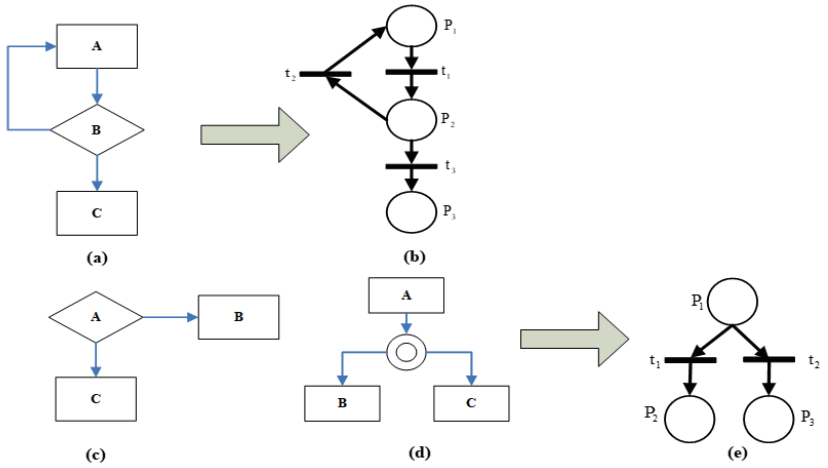
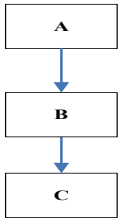
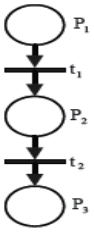
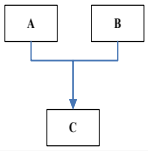
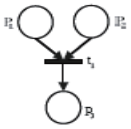
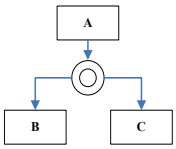
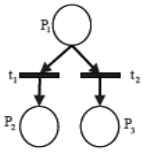
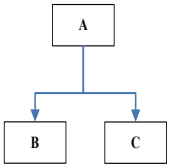
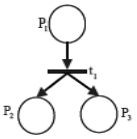
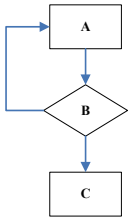
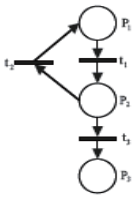


Fig. 11. (Color online) Petri nets representing the flow for the decision-making loop and selection.

Table 3
(Color online) Flow diagrams of five basic modules in Petri net.

Flow diagram	Petri net module	Petri net diagram	Description of application
	Buffer-type module		After P_1 is complete, operating program P_2 can be executed only through the action of a trigger firing at t_1. This method is used to complete all the operating procedures.
	AND-type module		P_1 and P_2 are completed concurrently before t_1 can be triggered to move.
	Event-conflict-type module		This module addresses event conflict during problem points occurring in operating procedures or during selection or decision-making requirements.
	Concurrent-execution-type module		Some operating procedures share the same type of scope and must be executed concurrently.
	Loop-token-type module		When an operating procedure executes an erroneous judgment, a loop is formed until the error is resolved, allowing the repeating operation to be stopped.

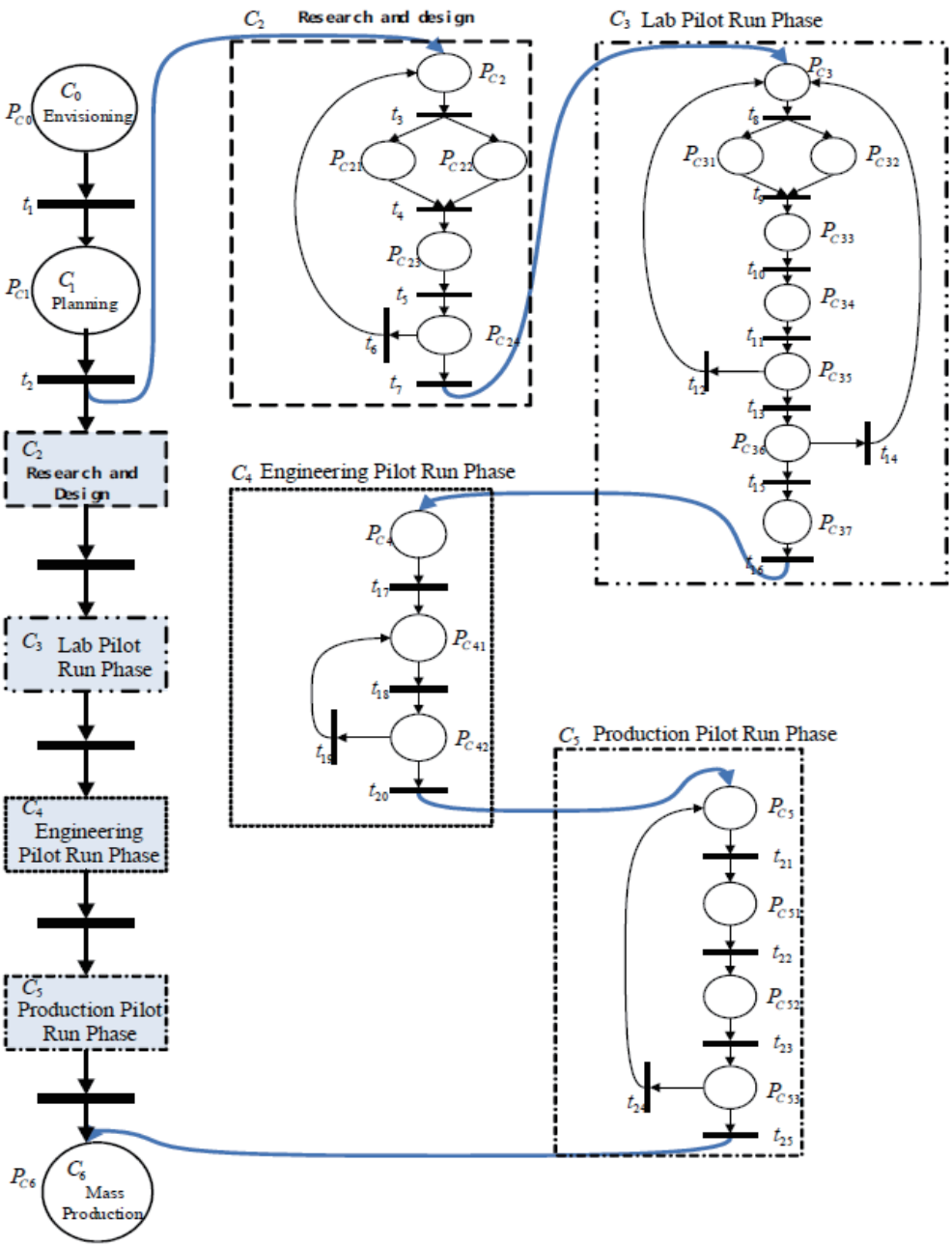


Fig. 12. (Color online) Petri net module for the flow of product development.

The hierarchy Petri nets method is used in this study to show the subsystems in C_2 (research and design phase), C_3 (lab pilot run phase), C_4 (engineering pilot run phase), and C_5 (production pilot run phase). In this way, the entire flow process for the product development model becomes clear and understandable. The module place and transition are explained in detail in Table 4.

Table 4
Representation of place and transition in module P_n in flow process of new product development.

Place representation			
Node	Explanation	Node	Explanation
P_{c0}	C_0 envisioning phase	P_{c1}	C_1 planning phase
P_{c2}	Determine specifications	P_{c21}	Circuit design and initial circuit diagram completed
P_{c22}	Mechanical general drawing completed	P_{c23}	Procurement based on preliminary material table for material cost estimation
P_{c24}	Review design feasibility during design review meeting	P_{c3}	Research and design (RD)
P_{c31}	Completion of formal circuit	P_{c32}	Completion of mechanical detail drawing
P_{c33}	Establish formal material table	P_{c34}	Prototyping
P_{c35}	Sample testing	P_{c36}	Send to customer testing
P_{c37}	Open local planning and response (LPR) meeting	P_{c4}	Verification of all sample components
P_{c41}	Extended producer responsibility (EPR)	P_{c42}	EPR testing
P_{c5}	Pedagogy and professional responsibility (PPR) preparation	P_{c51}	Preparation of standard operating procedures and scheduling
P_{c52}	Manufacturing	P_{c53}	PPR testing
P_{c6}	C_6 mass production		
Transition representation			
Node	Explanation	Node	Explanation
t_1	Organizer (project manager) starts a review meeting to identify new product ideas	t_2	PM-led review meeting explains the specifications and schedule for defining the development to various units
t_3	Project manager (PM) completes the product configuration list	t_4	Establish a preliminary list of materials and prepare sample testing materials
t_5	All materials should meet customer and company requirements regarding harmful substances	t_6	Re-evaluation of RD design
t_7	RD proposes product engineering spec for C_3 stage	t_8	Define exterior appearance of products and verify packaging methods
t_9	Enterprise manufacturing intelligence (EMI)/completion of security debugging and application	t_{10}	Prepare product pilot run material
t_{11}	Send samples to complete product design testing	t_{12}	Modify circuit debugging
t_{13}	Product assurance (PA) inspection passing requires test report to be completed	t_{14}	Reconfirm further changes
t_{15}	Complete the confirmation for the required number with the customer	t_{16}	Quality assurance (QA) completes focus type LPR test
t_{17}	Confirm normal or conditioned transition for special cases	t_{18}	Structure tooling completed and verified
t_{19}	PE and ME coordinate in assisting PD to solve production bugs, RD assists in solving problems related to design	t_{20}	QA completes detailed EPR test and PE guides arrangement of production operation process.
t_{21}	EMI/safety application passed	t_{22}	PPR material preparation completed
t_{23}	Manufacturing completed	t_{24}	Professional engineers (PE) and manufacturing engineers (ME) coordinate to assist PD to solve production errors, RD assists in solving problems in design
t_{25}	QA completes PPR test and mass production (MP)		

4. Conclusions and Recommendations

When teaching IT courses involving informal and formal logic, teachers must design activities to help students learn logical thinking with an established curriculum. This study explores how to use a Petri net to construct a logical modular system for IT teachers. A DES with a Petri net focuses on graphical description and is used for time Petri nets and hierarchical Petri nets. Toward improving IT courses, we investigated the use of a Petri net based on its advantages to develop a flow process module for designing IT courses. The new product development process is used to verify the feasibility of the module and its implementation.

To understand the structural properties of a Petri net and its dynamic behavior for application in new product development processes, we combined conventional and high-order Petri nets to develop a logical curriculum for IT courses. As a result, we developed a Petri net module in which places and transitions are defined. The results of this study are expected to provide a basic understanding of how to use a Petri net and DES to develop an educational curriculum, especially in IT courses. The module and its elements (places and transitions) are applied to the design and teaching of IT courses. The results also provide the basis for curriculum management using data collected with various sensing technologies, enabling teachers to monitor and evaluate student learning efficiency, thus enhancing the professionalism of teachers and the learning ability of students on IT courses. We verified the simplicity, easiness of understanding, powerful descriptive ability, and analytical ability of Petri nets for developing a deep learning model to meet education needs in collaboration with pedagogical experts.

References

1 Discrete Event Dynamic Systems: An Overview: https://repository.upenn.edu/cgi/viewcontent.cgi?article=1380&context=cis_reports (accessed March 2022).
2 E. Kučera, O. Haffner, P. Drahoš, R. Leskovský, and J. Cigánek: Appl. Sci. **10** (2020) 5027. https://doi.org/10.3390/app10155027
3 B. Carrascal: Proc. Int. Congr. Tools for Teaching Logic (TICTTL, 2011) 38–45.
4 A. Morou and N. Kalospyros: Proc. Congr. European Society for Research in Mathematics Education. (CERME, 2011) 1–9.
5 S. Butchart, T. Handfield, and G. Restall: Teach. Philos. **32** (2009) 1. https://doi.org/10.5840/teachphil20093212.
6 Logic in the Classroom, Four Activities: https://www.plato-philosophy.org/james-davis-logic-in-the-classroom-four-activities/ (accessed March 2022).
7 U. H. Engberg and G. Winskel: Brics Rep. Series **1** (2006) 176. https://doi.org/10.1007/3-540-58043-3_20
8 U. H. Engberg and G. Winskel: Proc. Colloq. Trees in Algebra and Programming (CAAP, 1990) 147–161.
9 H. H. Dang and B. Möller: Acta Inform. **52** (2015) 109. https://doi.org/10.1007/s00236-015-0216-3
10 R. Hamadi and B. Benatallah: Proc. 2003 Agile Development Conf. (ADC, 2003) 20–31.
11 S. Babaie, A. Khosrohosseini, and A. Khadem-Zadeh: J. Syst. Archit. **59** (2013) 582. https://doi.org/10.1016/j.sysarc.2013.06.004
12 J. Li, Z. Zhu, and X. Cheng: Complexity **2018** (2018) 1. https://doi.org/10.1155/2018/8261549
13 Kommunikation mit Automaten: http://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/ (in Germany) (accessed March 2022).
14 A. Bobbio: Proc. System Reliability Assessment (Springer, 1990) 103–143.
15 P. Bouvier, H. Garavel, and H. P. León: Petri Nets **2020** (2020) 3. https://dblp.org/rec/conf/apn/BouvierGL20.htm
16 M. Kucharik and Z. Balogh: Proc. Recent Developments in Intelligent Computing, Communication and Devices (ICCD, 2017) 1115–1124.
17 I. Grobelna and A. Karatkevich: Electronics **10** (2021) 2305. https://doi.org/10.3390/electronics10182305

18 A. Sadiq, F. Ahmad, S. A. Khan, and J. C. Valverde: Neural. Comput. Appl. **25** (2014) 1099. https://doi.org/10.1007/s00521-014-1590-4
19 K. L. Lo, H. S. Ng, and J. Trecat: IET Gener. Transm. Distrib. **144** (1997) 231. https://doi.org/10.1049/ip-gtd:19971060
20 X. Yi and K. J. Kochut: Proc. 2004 IEEE Int. Conf. Web Services (IEEE, 2004) 756. https://doi.org/10.1109/ICWS.2004.1314810
21 K. Jensen: Proc. Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS, 1997) 203–208.
22 R. Hamadi and B. Benatallah: Proc. 14th Australian Database Conf. (ADC, 2003) 191–200.
23 W. M. Zuberek: Microelectron. Reliab. **31** (1991) 627. https://doi.org/10.1016/0026-2714(91)90007-T
24 J. Dai, G. Su, Y. Sun, S. Ye, P. Liao, and Y. S. Yi: Proc. 2018 9th Int. Conf. E-Education, E-Business, E-Management and E-Learning (IC4E, 2018) 1–6.

About the Authors

Haidong Fu received his B.S. degree from Northeast Normal University, China, in 2004 and his M.Ed. from Northeast Normal University in 2006. Since 2018, he has been working as an associate professor at Changchun University, China, and in 2020, he enrolled as a Ph.D. candidate at Northeast Normal University. His research interests include education information, IT education, and teacher performance development. (fuhd.ccu@163.com)

Yueguang Xie received her B.S. and Ph.D. degrees from Northeast Normal University, China, in 1982 and 2007, respectively. Since 2002, she has been working as a professor at Northeast Normal University. Her research is related to IT education, information learning resources, high school technical curriculum, and education information in rural areas. (xieyg@nenu.edu.cn)

Jih-Fu Tu received his B.S. and M.S. degrees from Kaohsiung Normal University and Taiwan Normal University, Taiwan, in 1983 and 1989, respectively. He received his Ph.D. degree in science engineering from Preston University, USA, in 2003. From 1990 to 2020, he was a professor at the Electronic Engineering Department at St. John's University, US. Since 2021, he has been a professor at the Electrical Engineering Department of LungHwa University of Science and Technology, Taiwan. His research interests are education technologies, programmable circuit design, and IoT.
(tujihfu@gmail.com)