# Dynamic process modelling using Petri nets with applications to nuclear power plant emergency management

Madjid Tavana

Management Information Systems,
La Salle University, Philadelphia, PA 19141, USA
E-mail: tavana@lasalle.edu

**Abstract:** Complex information systems are difficult to model and expensive to build. Numerous process modelling methodologies, such as action workflow, data flow diagrams, decision trees, entity relationship diagrams, process maps, role activity diagrams and role interaction nets, have been developed to provide high level abstraction of a system. Despite their huge popularity, these methodologies cannot depict and verify the dynamic requirements of a system. In this study, we use Petri Nets (PNs) for dynamic process modelling of the emergency management system at a nuclear power plant. Decision trees used prior to PNs at the plant were inadequate for modelling complex emergency processes exhibiting sequential execution, conflict, concurrency, synchronisation, merging, confusion, or prioritisation. PNs with their graphical and precise nature and their firm mathematical foundation are especially useful in reducing the number of false evacuations at the plant.

**Keywords:** process modelling; PN; Petri net; dynamic systems; emergency management system; decision tree; nuclear power plant.

**Reference** to this paper should be made as follows: Tavana, M. (2008) 'Dynamic process modelling using Petri nets with applications to nuclear power plant emergency management', *Int. J. Simulation and Process Modelling*, Vol. 4, No. 2, pp.130–138.

**Biographical notes:** Madjid Tavana is a Professor of Management Information Systems and the Lindback Distinguished Chair of Information Systems at the La Salle University, where he served as the Chairman of the Management Department and Director of the Center for Technology and Management. He has been a Distinguished Faculty Fellow at the NASA's Kennedy Space Center, NASA's Johnson Space Center, Naval Research Laboratory – Stennis Space Center, and Air Force Research Laboratory. In 2005, he was awarded the prestigious Space Act Award by NASA. He holds an MBA, a PMIS, and a PhD in Management Information Systems. He received his Post-Doctoral Diploma in Strategic Information Systems from the Wharton School of the University of Pennsylvania. He has published in journals such as *Decision Sciences*, *Interfaces*, *Information Systems*, *Information and Management*, *Computers and Operations Research*, *Journal of the Operational Research Society*, and *Advances in Engineering Software*, among others.

## 1 Introduction

Information systems are complex artifacts that are difficult to model especially when the components of the system exhibit a variety of situations such as sequential execution, conflict, concurrency, synchronisation, merging, confusion, or prioritisation (Balduzzi et al., 2000; Mehrez et al., 1995). *Sequential execution* refers to the processing of precedence constraints; *conflict* refers to mutually exclusive activities or results; *concurrency* refers to simultaneous task operation; *synchronisation* refers to multiple resource usage in a single operation; *merging* refers to multiple precedence constraints; *confusion* refers to the combination of conflict and concurrency; and *prioritisation* refers to the determination of the priorities of activities. Lee et al. (2001) have argued the need for modelling techniques to verify and validate the reliability and quality of the processes in such complex systems.

Process modelling visually represents business processes by defining and depicting entities, activities, enablers and the relationships among them (Curtis et al., 1992). A business process is a set of related tasks performed to achieve a defined business outcome (Davenport and Short, 1990). Process models are widely used for decomposing organisational complexities, adapting best business practices, identifying process weaknesses, training end-users, and designing business blueprints (Peristeras and Tarabanis, 2000; Rosemann, 2000; Rosemann, 2006; Smith and Fingar, 2003; Scheer, 2000). Interest in business process modelling has grown increasingly over the past decade with some organisations conducting enterprise-wide and even global process modelling (i.e., Becker et al., 2005; Scheer et al., 2003). The literature also reports on a wide range of business applications in activity based

cost management, customer relationship management, knowledge management, operations management, supply chain management, total quality management, workflow management and simulation model management (Becker et al., 2005; Kalpic and Bernus, 2006; Kis et al., 2000; Rosemann, 2000; Rosemann and Zur Mühlen, 1997; Viswanadham and Srinivasa Raghavan, 2000).

Numerous process modelling methodologies such as action workflow diagrams, data flow diagrams, decision trees, entity relationship diagrams, flowcharts, general process charts, integrated definition of function modelling, process activity charts, process maps, quality function deployment, role activity diagrams, role interaction net and simulation have been developed to provide high level abstraction of a system. Many of these methodologies have their roots in a variety of disciplines such as coordination theory, database design, software engineering and system analysis. As a result, each method models business processes from different perspectives and has different features. These methodologies emphasise the description of 'what' or the functional aspects of the system. However, a second dimension of modelling complexity, control flow, emphasising the description of 'how', is often as important as functional specifications (Yourdon, 1989). Most of these methodologies are inadequate for modelling concurrent and asynchronous systems (Murata, 1984). In addition, these methods are generally based on practice-driven constructs and utilise graphical notation to represent these constructs. Consequently, most of them lack formal theoretical foundations and mechanisms for verifying their 'correctness' (i.e., their completeness, consistency, and feasibility). Several authors have tried to address the control logic problem inherent in process modelling with Petri Nets (PNs) (Bullers, 1991; Richter and Maffeo, 1993; Tavana et al., 2003; van Hee et al., 1991). In the next section, we present a short literature review on PNs followed by the PN principles and formalism in Section 3. Section 4 illustrates the application of PNs to emergency management process modelling at a nuclear power plant and Section 5 presents discussions and conclusions.

## 2 Literature review on PNs

PNs are well-suited for the design, specification, and formal verification of complex information systems (Sakthivel and Tanniru, 1988–1889). PNs with their graphical and precise nature and their firm mathematical foundation are commonly used to model many complex systems. Their graphical nature allows for models that are easy to understand while their formal semantics allow for precise and unambiguous descriptions. PNs are particularly considered a richer, more versatile and dynamic graphical tool in the development and validation of business systems (Mehrez et al., 1995; Wong, 2001).

PNs were initially defined by Petri (1962) and later refined and named after him by Holt (1971). Peterson (1981) elegantly discusses the dynamic behaviour of PNs, while Murata's tutorial review paper provides a thorough review of PNs' history and applications (Murata, 1989). PNs and their modifications provide a rich and versatile approach to modelling. They have been proven to be useful for the modelling and analysis of several classes of systems including web-based systems (Huang et al., 2008; Zhovtobryukh, 2007), simulation systems (Piera et al., 2004), communication systems (Berthelot and Terrat, 1982), knowledge-based systems (Huang et al., 2008; Jantzen, 1980), and process control systems (Bruno and Marcchetto, 1986; Camurri et al., 1993; Zave, 1982).

As a graphical tool, PNs provide a visual medium for a modeller to describe a complex system. As a mathematical tool, PN models can be represented by linear algebraic equations, creating the possibility for the formal analysis of the model (Zurawski and Zhou, 1994). Mathematical properties of PN can be classified into

- structural properties that depend on the net structure
- behavioural properties that depend on the initial and subsequent markings.

Mathematically, analyses of PNs can be based on enumerating all possible markings to form reachability trees and/or through methods and theories in discrete mathematics like matrix equations. We use both graphical and mathematical properties of PN in process modelling. As a graphical tool, PNs are used to enhance communications and produce accurate and complete specifications while the mathematical properties of PNs are used to detect deadlock, overflow, and irreversible situations. Performance evaluation is also possible through mathematical analysis of the model using stochastic timed PNs.

One of the strengths of PNs is their broad based applicability to a wide range of systems. Ordinary place-transition PNs are used in this study for process modelling. Place-transition nets are classical models with black tokens that model the control flow in business systems quite comfortably and provide efficient ways of 'qualitative' verification. In addition to their modelling power, ordinary PNs are described as both a graphical and mathematical tool. We show how PNs provide process and control specifications that relate the descriptions of 'what' and 'how' more closely to the actual system implementation. In addition to ordinary PNs, timed PNs, stochastic PNs, coloured PNs, and fuzzy PNs are also widely used to model business systems. Timed PNs are those with places or transitions that have time durations in their activities (Liu et al., 2007). Stochastic PNs include the ability to model randomness in a situation, and also allow for time as an element in the PN (Murata, 1989; Lee et al., 2001). Coloured PNs allow the user and developer to witness the changes in places and transitions through the application of colour-specific tokens, and movement through the system can be represented through the changes in colours (Chen et al., 2001). Fuzzy PNs are used to model fuzzy rule-based reasoning to handle uncertain and imprecise information (Huang et al., 2008; Li and Lara-Rosano, 2000).

PNs are especially useful for systems that may possess concurrent, distributed, asynchronous, parallel, or event-driven qualities. We illustrate PNs and their suitability and effectiveness to model a nuclear power plant emergency management system. The remainder of the paper is organised as follows.

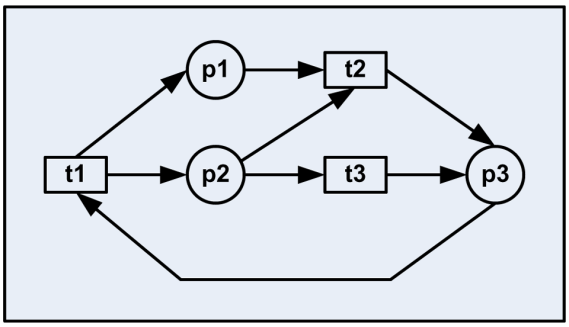
## 3 PN principles and formalism

Classical PNs as defined by Petri (1962) and further discussed by Peterson (1981) are identified by 5-tuples $(P, T, I, O, \mu)$ where $P = \{p_1, p_2, ..., p_m\}$ is a set of places; $T = \{t_1, t_2, ..., t_n\}$ is a set of transitions; $[I \subseteq PxT]$ is the input function from $P$ to $T$; $[O \subseteq TxP]$ is the output function from $T$ to $P$; and $\mu$, called marking, is a function that defines a mapping from a set of places $P$ to $Z$ (here $Z$ denotes the set of all nonnegative integers).

$$\mu : P \to Z \quad \text{where} \quad \mu_i = \mu(p_i) \in Z, p_i \in P \quad i = \{0, 1, ..., m\}.$$

Mathematically, a PN is a directed bipartite graph with two different types of node called *places* and *transitions*. A place $p$ is presented with a circle and a transition $t$ is presented by a rectangle. The nodes are connected through directed *arcs*. Directed arcs from $p$ to $t$ create *input places* while directed arcs from $t$ to $p$ create *output places*. Input places are a set of places that can fire a transition, while output places are a set of places that are associated with the results (outputs) from a transition. Only the static properties of a system are presented by a PN structure, and dynamic system properties result from PN execution. Execution requires the use of tokens or markings (denoted by dots) associated with places. Each place contains zero or many tokens drawn as black dots. The execution of a PN may affect the number of tokens in a place. A transition is called *enabled* when each of its input places has enough tokens. A transition can be *fired* only if it is enabled. When a transition is fired, tokens from input places are used to produce tokens in output places. We will use the PN shown in Figure 1 to illustrate the classical PN.

**Figure 1** Simple Petri Net example (see online version for colours)




With one token placed in $p_1$ and two tokens placed in $p_2$, transitions $t_2$ and $t_3$ are enabled. Firing $t_2$ and $t_3$ consumes three tokens (one from $p_1$ and two from $p_2$) and produces two tokens in $p_3$. Now transition $t_1$ is enabled. Firing $t_1$ consumes one of the two tokens in $p_3$ (leaving $p_3$ with one token) and produces two tokens, one in $p_1$ and one in $p_2$. Mathematical properties of the PN shown in Figure 1 are presented next.

The PN in Figure 1 is constructed with three places $(P = \{p_1, p_2, p_3\})$ and three transitions $(T = \{t_1, t_2, t_3\})$. The input and output mapping of this PN is given as:

$$I(t_1) = \{p_3\} \qquad O(t_1) = \{p_1, p_2\}$$

$$I(t_2) = \{p_1, p_2\} \qquad O(t_2) = \{p_3\}$$

$$I(t_3) = \{p_2\} \qquad O(t_3) = \{p_3\}.$$

Any PN can be specified in matrix form as a *D*-Matrix with *m* rows and *n* columns, where *m* is the number of *transitions* and *n* is the number of *places* in the PN. For each position [*i*, *j*] in the matrix, a 1 is placed in the position if transition *i* receives input from place *j*. A 0 is placed if transition *i* does not receive input from place *j*:

$$D- = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Similarly a *D*+ Matrix with *m* rows and *n* columns can be constructed, where *m* is the number of *transitions* and *n* is the number of *places* in the PN. For each position [*i*, *j*] in the matrix, a 1 is placed in the position if transition *i* produces output to place *j*. A 0 is placed if transition *i* does produce output to place *j*:

$$D+ = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

The Composite Change Matrix (Matrix *D*) can be computed by subtracting *D*– from *D*+:

$$[D+] - [D-] = [D]$$

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ -1 & -1 & 1 \\ 0 & -1 & 1 \end{bmatrix}.$$

In addition, a $1 \times m$ matrix, representing the firing of the PN is constructed. In each position [1, *j*] places the number of times transition *j* is to fire. The Transition Matrix for our PN is:

Transition Matrix = $\begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$ ($t_2$ and $t_3$ firing because of the tokens in $p_1$ and $p_2$).

Finally, a $1 \times n$ matrix is constructed showing the current marking of the PN. In each position [1, *j*], the identified number of tokens in position *j* are placed. The Marking Matrix for our PN is:

Marking Matrix = $\begin{bmatrix} 1 & 2 & 0 \end{bmatrix}$ (given one token in $p_1$ and two tokens in $p_2$).

The marking of the PN after the transition specified in the Transition Matrix (next marking) can be found as:

$$([\text{Transition Matrix}][D] + [\text{Marking Matrix}] = [\text{Next Marking}]$$

$$\begin{bmatrix}0 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 1 & -1\\ -1 & -1 & 1\\ 0 & -1 & 1\end{bmatrix} + \begin{bmatrix}1 & 2 & 0\end{bmatrix} = \begin{bmatrix}0 & 0 & 2\end{bmatrix}.$$

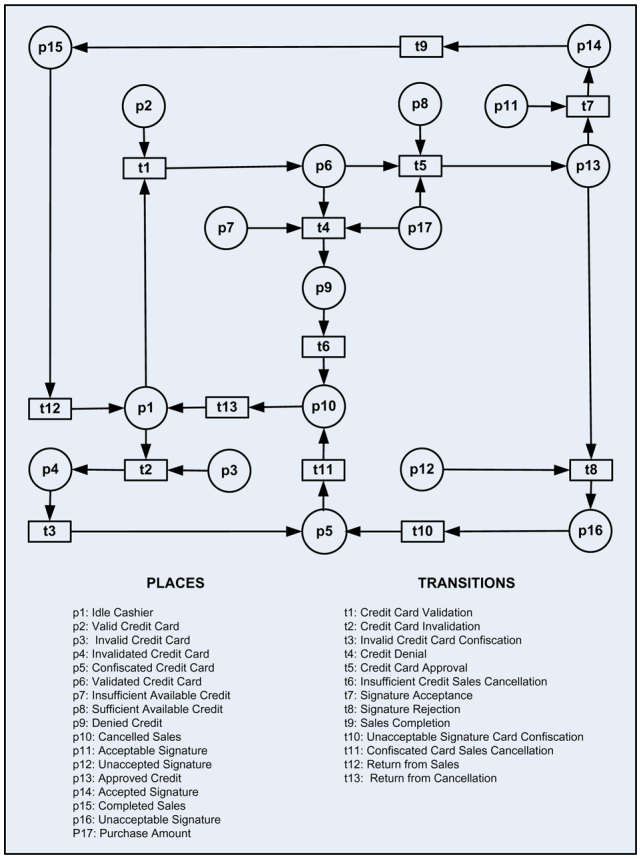
From a modelling perspective, PNs can be characterised as a "conceptual model with analytical qualities". A 'conceptual model' generally represents a graphical approach, while an 'analytical model' expresses functional and mathematical relationships (Mehrez et al., 1995). As a hybrid modelling approach, PNs can display several important properties, including the ability to model situations for simulation analysis and to describe system behaviour (Kim et al., 2001). These abilities are useful in modelling complex business processes.

Next, we present a simple credit card purchase processing problem. In this problem, when an invalid credit card is presented to a cashier, the sales transaction is cancelled and the cashier is required to confiscate the credit card. If the credit card is valid but there is insufficient credit to pay for the purchase, the sales transaction is cancelled. When there is sufficient credit available to pay for the purchase, the cashier is required to verify the customer's signature. If the signature is verified, a sales transaction is completed. If the signature is not verified, the sales transaction is cancelled and the cashier is required to confiscate the credit card. Most process modelling methodologies are static and cannot represent sequential execution, conflict, concurrency, synchronisation, merging, confusion, or prioritisation. Figure 2 shows the credit card purchase processing problem represented with a PN.

As it is shown in Figure 2, the credit card purchase processing PN can be represented with 17 places and 13 transitions. Only the static properties of the system are presented by the PN structure presented in Figure 2. The dynamic properties of this PN can be examined and verified with the user(s) through execution which requires the use of tokens associated with places. Each place contains zero or many tokens. A token placed in $p_1$ indicates an *Idle Cashier* state and the system waits for a signal from $p_2$ or $p_3$. A token placed in $p_2$ signifies a *Valid Card* while a token placed in $p_3$ signifies an *Invalid Card*. With one token placed in $p_1$ and $p_2$; transition $t_1$, *Card Validation* is enabled. With one token placed in $p_1$ and $p_3$; transition $t_2$, *Card Invalidation* is enabled.

Firing $t_2$ consumes the two tokens in $p_1$ and $p_3$ and produces one token in $p_4$, representing the *Invalid Card* state. The one token in $p_4$ enables transition $t_3$, *Invalid Card Confiscation*. Firing $t_3$ produces one token in $p_5$, the *Confiscated Card* state which in turn enables transition $t_{11}$, *Confiscated Card Sales Cancellation*. Firing $t_{11}$ places a token in $p_{10}$, the *Cancelled Sales* state which in-turn enables transition $t_{13}$, *Return from Cancellation*. Firing transition $t_{13}$, produces one token in $p_1$ and the system returns to its initial *Idle Cashier* state.

**Figure 2** Credit card purchase processing Petri Net diagram and notations (see online version for colours)




Firing $t_1$ consumes the two tokens in $p_1$ and $p_2$ and produces one token in $p_6$, representing the *Valid Card* state. The system waits for a signal from $p_7$ or $p_8$. A token placed in $p_7$ signifies *Insufficient Available Credit*, while a token placed in $p_8$ signifies *Sufficient Available Credit*. With one token placed in $p_6$ and $p_7$; transition $t_4$, *Credit Denial*, is enabled. With one token placed in $p_6$ and $p_8$ transition $t_5$, *Credit Approval*, is enabled. As shown in this example, tokens in PNs can represent sequential and concurrent execution of several transactions. In addition, the movement of the tokens throughout the net can be synchronised and prioritised. Finally, the execution of a PN can point to merging and confusion problems.

There are many frameworks, methods, and tools to build and validate information systems. Some are more suited to transaction processing systems. According to van Hee et al. (1991), designing a classical information system means automating an existing, well-defined manual system (or set of processes). However, when designing dynamic information systems, it is unwise to automate the existing processes completely, since many of these processes rely on heuristics and are not easily replicated by a machine. Many dynamic information systems have failed as a result of the designers' inability to differentiate between those processes that should be automated and those that should remain manual (van Hee et al., 1991).

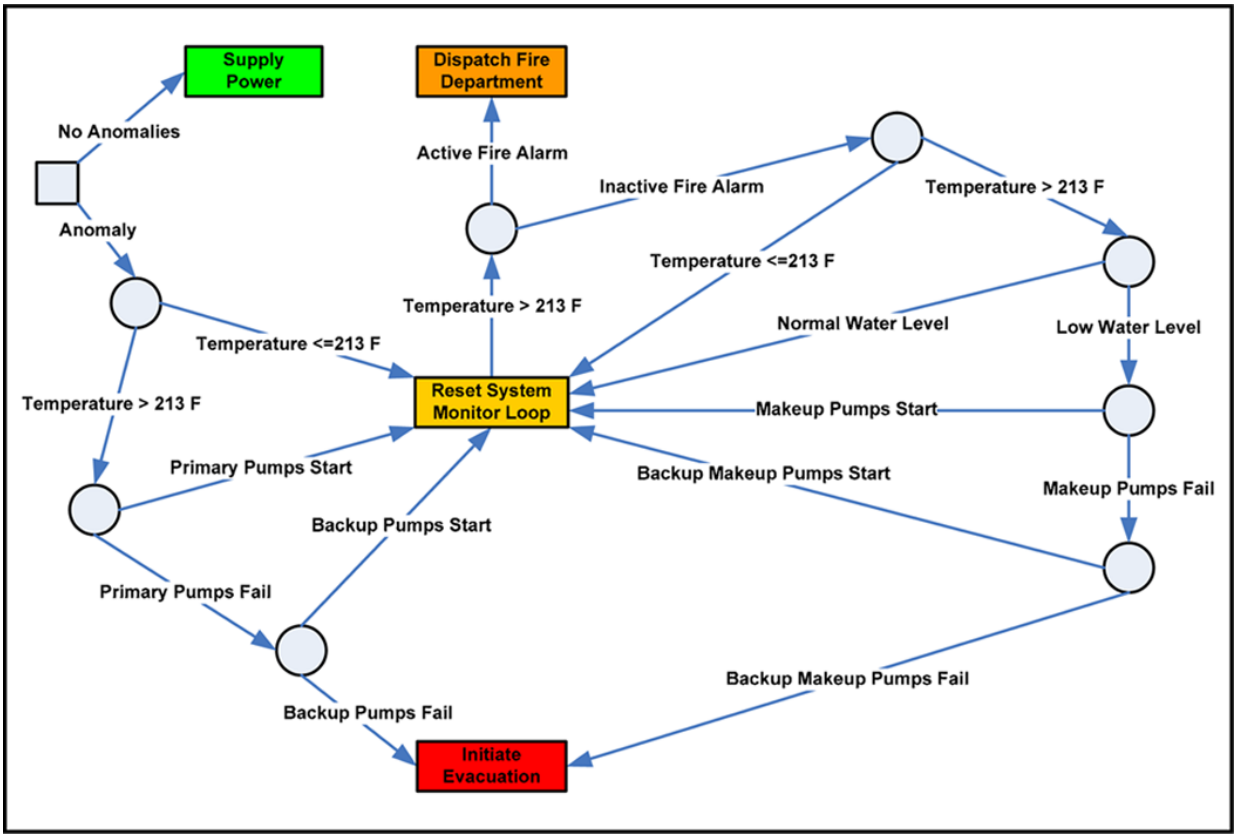
## 4 Application problem

In this section, we use PNs to model the emergency management system at the Khaskovo[1] nuclear power plant in Eastern Europe. The Khaskovo PN system was implemented using Petri Maker in a visual environment (Mortensen, 2003). For illustration purposes and ease of reading, we have demonstrated the emergency management process with the decision tree presented in Figure 3. The evaluation of the tree begins at the root (shown as a square) and branches leading to possible events (shown as circles). At the end of the tree, each path leads to an outcome (shown as a rectangle). Trees may be extended for more than one time period, but cannot represent concurrent events or dynamic behaviours and after only a few repetitions, they may grow into hundreds of nodes (Cooper et al., 2007).

Although decision trees are popular and easy to use, they are not suitable for modelling sequential execution, prioritisation, and concurrency since multiple decision processes cannot be traced by the same decision tree at the same time. In addition, decision trees are static and cannot illustrate dynamic behaviour such as conflict, merging, and confusion. Decision trees represent a precedence ordering among events, and constrain those events to occur only after some other events have occurred. The occurrence of non-sequential events (i.e., concurrent events) are incomparable with decision trees and other popular process modelling techniques such as action workflow diagrams, data flow diagrams, process maps, and role activity diagrams. Decision trees were used in the past to model the emergency management system at Khaskovo. However, there were several incidents where the system modelled with decision trees failed to respond to concurrent faults resulting in unnecessary evacuations.

A team of nine Emergency Systems Operators (ESOs) was selected to participate in the design and development of a new emergency management system capable of handling concurrent faults simultaneously. The team of expert ESOs held several meetings to discuss the emergency management process at Khaskovo. After several rounds of group discussions, there was a consensus among the ESOs that the procedure described next should be followed by all ESOs in response to emergencies at the plant.

**Figure 3** Khaskovo's emergency management decision tree (see online version for colours)




The ESOs respond to various types of emergencies. If there are no general anomalies or alarms, the ESO continues to supply power to the grid. Upon an anomaly or alarm, the ESO is expected to check the core temperature. The core temperature must remain at the normal level of 213°F. If the core temperature inspection shows normal levels, the ESO resets and restarts the system monitor loop. However, if the core temperature exceeds 213°F, the ESO activates the primary cooling water pumps. If the primary cooling water pumps start successfully, the ESO resets and restarts the system monitor loop. However, if the primary cooling water pumps fail to start, the ESO activates the backup pumps. If the backup pumps start successfully, the ESO resets and restarts the system monitor loop.

If the backup pumps fail to start, the ESO initiates the evacuation process, as a core breech is imminent.

The process of reset and restart of the system monitor loop involves the following procedures. After every reset and restart of the system, the ESO verifies the core temperature. If the core temperature verification shows normal levels of 213°F or less, the ESO resets and restarts the system monitor loop. If the core temperature is greater that 213°F, the ESO checks the fire alarm. If the fire alarm is not activated, the ESO verifies the core temperature again. If the fire alarm is activated, the ESO dispatches the fire department. If the fire department can keep the temperature at 213°F or less, the ESO resets and restarts the system monitor loop. If the core temperature is increased beyond the normal level of 213°F, the ESO checks the core water level. If the core water level is normal, the ESO resets and restarts the system monitor loop. However, if the core water level is low, the ESO is expected to activate the makeup water pumps. If the makeup water pumps start successfully, the ESO resets and restarts the system monitor loop. However, if the makeup water pumps fail to start, the ESO must activate the backup makeup water pumps. If the backup makeup water pumps start successfully, the ESO resets and restarts the system monitor loop. If the backup makeup water pumps fail to start, the ESO initiates the evacuation process. Figures 4(a) and (b) shows the emergency management system at the Khaskovo modelled by a PN with 37 places and 26 transactions.

Only the static properties of the system are presented by the PN structure presented in Figure 4(a). The dynamic properties of the emergency management PN at Khaskovo can be examined and verified with the ESOs through execution which requires the use of tokens associated with places. The PN presented in Figure 4(a) is especially useful for modelling concurrent, distributed, asynchronous, or parallel processes and removing confusions and bottlenecks in the system through simulation and user verification.

An evaluation of the new emergency management system showed a significant drop in the number of false evacuations at Khaskovo. The data collected over a 42-month period prior to the implementation of the new system showed an average of 2.68 false evacuations per month. However, the data collected over a 12-month period after the implementation of the new system showed an average of 0.42 false evacuations per month. Test of means between the pre and post emergency management system implementation shows that the 2.26 difference in the average number of false evacuations is statistically significant ($\alpha = 0.05$). In summary, this study revealed a significant drop in 'wrong' evacuations after the implementation of the new emergency management systems developed with PN process modelling proposed in this study.

**Figure 4(a)** Khaskovo's emergency management Petri Net (see online version for colours)

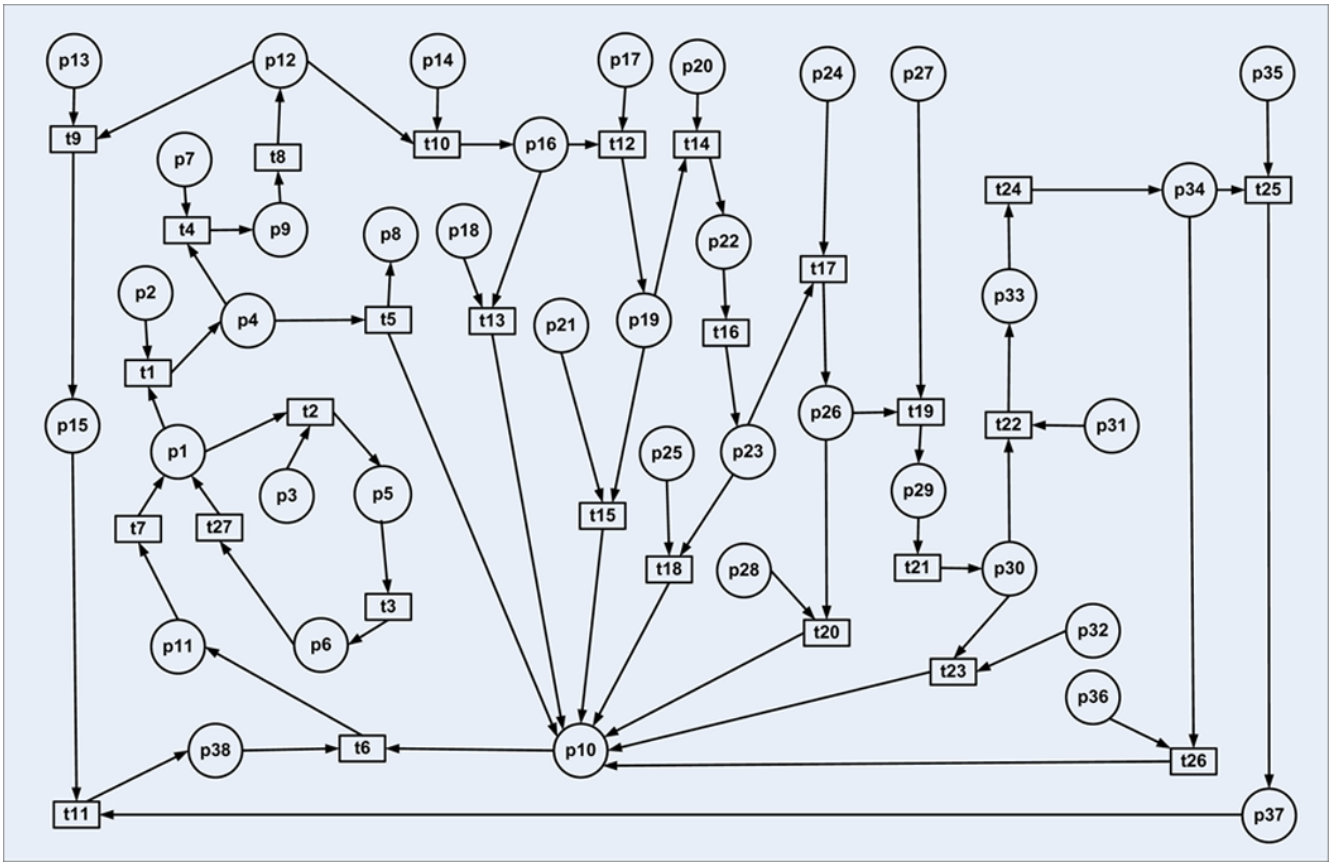


**Figure 4(b)** Khaskovo's emergency management Petri Net notations (see online version for colours)


## 5 Discussions and conclusions

The power of PNs is their interactive mode which allows a real-time observation and analysis of the system. Tokens in PNs can represent sequential and concurrent execution of several transactions. Most process modelling techniques (i.e., decision trees, data flow diagrams, process maps) are static and cannot represent multiple and concurrent execution of events. In addition, the movement of the tokens through the PNs can be synchronised and prioritised. Finally, the execution of a PN can point to merging and confusion problems. We illustrate some of these behaviours by referring to a series of simple examples in Figure 4(a).

*Sequential execution* refers to precedence constraints where an activity must be completed before another could get started. For example, backup pumps failure verification ($t_9$) cannot start unless the primary cooling water pumps are active ($t_8$). Although traditional process modelling methods can model sequential execution, most of them cannot represent control flow of multiple faults in the emergency management systems at Khaskovo. *Conflict* refers to mutually exclusive activities. For example, the system may exhibit either the presence of high temperature ($t_4$) or normal temperature ($t_5$). Again, the execution of a PN with multiple tokens can point to the system's inability in resolving conflict. *Concurrency* refers to simultaneous task operation. For example, one token in the PN can point to active fire alarms ($t_{14}$) while another points to active makeup water pumps ($t_{21}$). *Synchronisation* refers to multiple resource usage in a single operation. For example, it is possible for two tokens in the system, each representing an independent emergency, signaling dispatching of the fire department ($t_{16}$) to two different locations. *Merging* refers to multiple precedence constraints. For example, both backup pumps ($t_9$) and backup makeup water pumps ($t_{25}$) must fail before an evacuation process is initiated ($t_{11}$). *Prioritisation* refers to the determination of the priorities of activities. For example, one token in the PN can indicate active fire alarms ($t_{14}$) while another indicate active makeup water pumps ($t_{21}$). Utilising the prioritisation properties of PNs, active fire alarms ($t_{14}$) can be given higher priority.

Business information systems are complex artifacts that are difficult to design, develop, and validate. Various techniques such as decision trees, data flow diagrams, entity relationship diagrams, process maps, role activity diagrams and role interaction net are commonly used for process modelling. However, these static approaches cannot replicate the dynamic behaviour of systems with sequential execution, conflict, concurrency, synchronisation, merging, confusion, or prioritisation.

In this study, we showed that PNs have great potential for providing high-level abstraction in the systems development life cycle. PNs are especially suitable for modelling and simulation of complex systems requiring user verification and validation.

In summary, PN models provide a powerful modelling tool for representing information and control flows. PN models possess a meta-model capability to replace alternative process modelling frameworks. This meta-model capability provides the PNs with the modelling versatility required to represent complex systems not easily modelled by traditional frameworks. We should also note that the principal limitation of PNs is their inability to replace the traditional algorithmic models such as linear, non-linear, and dynamic programming. Other limitations of PNs include their limited capability for hierarchical modelling (using sub-models within a model) and extensibility (defining higher level models from lower level sub-models).

## Acknowledgements

The author is indebted to the Editor and the anonymous reviewers for their comments and constructive suggestions.

## References

Balduzzi, F., Giua, A. and Menga, G. (2000). 'First-order hybrid petri nets: a model for optimization and control', *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 4, pp.382–399.

Becker, J., Kugeler, M. and Rosemann, M. (2005) *Process Management*, 2nd ed., Springer-Verlag, Berlin, Germany.

Berthelot, G. and Terrat, R. (1982) 'Petri-net theory for correctness of protocols', *IEEE Transactions on Communication*, Vol. 30, No. 12, pp.2497–2505.

Bruno, G. and Marchetto, G. (1986) 'Process-translatable petri nets for the rapid prototyping of process control systems', *IEEE Transactions on Software Engineering*, Vol. 12, No. 2, pp.346–357.

Bullers, W.I. (1991) 'A tripartite approach to information systems development', *Decision Sciences*, Vol. 22, No. 1, pp.120–135.

Camurri, A., Franchi, P., Gandolfor, F. and Zaccaria, R. (1993) 'Petri net based process scheduling: a model of the control system of flexible manufacturing systems', *Journal of Intelligent and Robotic Systems*, Vol. 8, No. 1, pp.99–123.

Chen, S.C., Ke, Y.L. and Wu, J.S. (2001) 'Coloured petri-nets approach for solving distribution system contingency', *Proceedings of the IEEE*, Vol. 148, No. 5, pp.463–470.

Cooper, K., Brailsford, S.C. and Davies, R. (2007) 'Choice of modelling technique for evaluating health care interventions', *Journal of the Operational Research Society*, Vol. 58, pp.168–176.

Curtis, B., Keller, M.I. and Over, J. (1992) 'Process modelling', *Communications of ACM*, Vol. 35, No. 9, pp.75–90.

Davenport, T.H. and Short, J.E. (1990) 'The new industrial engineering: information technology and business process redesign', *Sloan Management Review*, Vol. 31, No. 4, pp.11–27.

Holt, A.W. (1971) 'Introduction to occurrence systems', in Jacks, L. (Ed.): *Associate Information Techniques*, American Elsevier, New York, pp.175–203.

Huang, Y-M., Chen, J-N., Huang, T-C., Jeng, Y-L. and Kuo, Y-H. (2008) 'Standardized course generation process using dynamic fuzzy petri nets', *Expert Systems with Applications*, Vol. 34, pp.72–86.

Jantzen, M. (1980) *Structures Representation of Knowledge by Petri Nets as an Aid for Teaching and Research*, *Lecture Notes in Computer Science*, Springer Verlag, Berlin.

Kalpic, B. and Bernus, P. (2006) 'Business process modelling through the knowledge management perspective', *Journal of Knowledge Management*, Vol. 10, No. 3, pp.40–56.

Kim, C.H., Yim, D.S. and Weston, R.H. (2001) 'An integrated use of IDEF0, IDEF3 and petri-net methods in support of business process modelling', *Proceedings of the Institution of Mechanical Engineers*, Vol. 215, No. 4, pp.317–329.

Kis, T., Kiritsis, D. and Xirouchakis, P. (2000) 'A Petri net model for integrated process and job shop production planning', *Journal of Intelligent Manufacturing*, Vol. 11, pp.191–207.

Lee, J., Pan, J.I. and Kuo, J.Y. (2001) 'Verifying scenarios with time petri-nets', *Information and Software Technology*, Vol. 43, No. 13, pp.769–781.

Li, X. and Lara-Rosano, F. (2000) 'Adaptive fuzzy Petri nets for dynamic knowledge representation and inference', *Expert Systems with Applications*, Vol. 19, No. 3, pp.235–241.

Liu, R., Kumar, A. and van der Aalst, W. (2007) 'A formal modelling approach for supply chain event management', *Decision Support Systems*, Vol. 43, pp.761–778.

Mehrez, A., Muzumdar, M., Acar, W. and Weinroth, G. (1995) 'A petri-net model view of decision making: an operational analysis', *Omega – The International Journal of Management Science*, Vol. 23, No. 1, pp.63–78.

Mortensen, K.H. (2003) *Petri Nets Tools and Software*, http://www.daimi.au.dk/PetriNets/tools

Murata, T. (1984) 'Modelling and analysis of concurrent systems', in Vicks, C.R. (Ed.): *Handbook of Software Engineering*, Van Nostrand Reinhold, New York, pp.39–63.

Murata, T. (1989) 'Petri nets: properties, analysis and applications', *Proceedings of the IEEE*, 77, Vol. 4, pp.541–580.

Peristeras, V. and Tarabanis, K. (2000) 'Towards an enterprise architecture for public administration using a top-down approach', *European Journal of Information Systems*, Vol. 9, pp.252–260.

Peterson, J.L. (1981) *Petri Net Theory and the Modelling of Systems*, Prentice-Hall, Inc., Morristown, NJ.

Petri, C.A. (1962) *Kommunikation mit Automaten*, University of Bonn, English Translation by Greene, C.F. (1965) *Communication with Automata*, Supplement to Technical Report RADC-TR-65-377, Vol. 1, Rome Air Development Center, Griffith Air Force Base, Rome, NY.

Piera, M.A., Narciso, M., Guasch, A. and Riera, D. (2004) 'Optimization of logistic and manufacturing systems through simulation: a colored petri net-based methodology', *Simulation*, Vol. 80, No. 3, pp.121–129.

Richter, G. and Maffeo, B. (1993) 'Toward a rigorous interpretation of ESNL – extended systems modelling language', *IEEE Transaction on Software Engineering*, Vol. 19, No. 2, pp.165–181.

Rosemann, M. (2000) 'Using reference models within the enterprise resource planning life cycle', *Australian Accounting Review*, Vol. 3, No. 22, pp.19–31.

Rosemann, M. and Zur Mühlen, M. (1997) 'Evaluation of workflow management systems – a meta model approach', *Australian Journal of Information Systems*, Vol. 6, No. 1, pp.103–116.

Rosemann, M. (2006) 'Potential pitfalls of process modeling: part a', *Business Process Management Journal*, Vol. 12, No. 2, pp.249–254.

Sakthivel, S. and Tanniru, M.R. (1988–1889) 'Information verification and validation during requirement analysis using Petri nets', *Journal of Management Information Systems*, Vol. 5, No. 3, pp.33–52.

Scheer, A-W. (2000) *ARIS – Business Process Frameworks*, 3rd ed., Springer-Verlag, Berlin, Germany.

Scheer, A-W., Abolhassan, F., Jost, W. and Kirchmer, M. (Eds.) (2003) *Business Process Change Management: ARIS in Practice*, Springer-Verlag, Berlin, Germany.

Smith, H. and Fingar, P. (2003) *Business Process Management*, The third wave, Meghan-Kiffer Press, Tampa, FL.

Tavana, M., Ortiz, J. and Torney, S. (2003) 'Modelling station duty officer operations assistant at Johnson space center', *Advances in Engineering Software*, Vol. 34, No. 3, pp.139–162.

van Hee, K.M., Somers, L.J. and Voorhoeve, M. (1991) 'A modelling environment for decision support systems', *Decision Support Systems*, Vol. 7, pp.241–251.

Viswanadham, N. and Srinivasa Raghavan, N.R. (2000) 'Performance analysis and design of supply chains: a petri net approach', *Journal of the Operational Research Society*, Vol. 51, No. 10, pp.1158–1169.

Wong, M.L. (2001) 'A flexible knowledge discovery system using genetic programming and login grammars', *Decision Support Systems*, Vol. 31, pp.405–428.

Yourdon, E. (1989) *Modern Structured Analysis*, Prentice Hall, Englewood Cliffs, New Jersey.

Zave, P. (1982) 'An operational approach to requirements specification for embedded systems', *IEEE Transactions on Software Engineering*, Vol. 8, No. 3, pp.250–269.

Zhovtobryukh, D. (2007) 'A petri net-based approach for automated goal-driven web service composition', *Simulation*, Vol. 83, No. 1, pp.33–63.

Zurawski, R. and Zhou, M-C. (1994) 'Petri nets and industrial applications: a tutorial', *IEEE Transactions on Industrial Electronics*, Vol. 41, No. 6, pp.567–583.

## Note

[1]The name of the nuclear power plant is changed to protect the anonymity of the facility.