Quantitative evaluation of Pandora Temporal Fault Trees via Petri Nets

Sohag Kabir, Martin Walker, and Yiannis Papadopoulos

Department of Computer Science, University of Hull, Hull, UK
(e-mail:{ s.kabir@2012., martin.walker@, y.i.papadopoulos@ } hull.ac.uk).

Abstract: Using classical combinatorial fault trees, analysts are able to assess the effects of combinations of failures on system behaviour but are unable to capture sequence dependent dynamic behaviour. Pandora introduces temporal gates and temporal laws to fault trees to allow sequence-dependent dynamic analysis of events. Pandora can be easily integrated in model-based design and analysis techniques; however, the combinatorial quantification techniques used to solve classical fault trees cannot be applied to temporal fault trees. Temporal fault trees capture state and therefore require a state space solution for quantification of probability. In this paper, we identify Petri Nets as a possible framework for quantifying temporal trees. We describe how Pandora fault trees can be mapped to Petri Nets for dynamic dependability analysis and demonstrate the process on a fault tolerant fuel distribution system model.

Keywords: Dependability Analysis, Fault Tree Analysis, Stochastic Petri Nets, Temporal Fault Trees.

1. INTRODUCTION

Our dependence on increasingly complex safety critical systems has made the dependability of such systems a prime concern for modern society. System safety and reliability are two key aspects of system dependability, and Fault Tree Analysis (FTA) is a well-established and widely used analysis method for evaluating these two properties. In FTA, analysis starts with a top event (system failure) and iteratively works backward to determine the root causes of system failure. To show logical connections between different faults and their causes, fault trees utilise a graphical representation based on Boolean logic (Vesely et al., 2002). Qualitative analysis is performed by reducing them to minimal cut sets (MCS), which are the smallest combinations of basic events (i.e., leaf nodes of the tree) that are necessary and sufficient to cause the top event. Quantitative analysis can estimate the unreliability of the system from probabilistic data about basic events, which typically represent component failures.

In dynamic systems with mode and state changes, accurately capturing failure behaviour requires understanding the order in which events occur. Dynamic Fault Trees (DFTs) (Dugan, Bavuso and Boyd, 1992) and Pandora temporal fault trees (TFTs) (Walker, 2009) are two FTA extensions that capture dynamic behaviour. DFTs are typically used as a quantitative method and are analysed via conversion into Markov chains. Pandora introduces temporal gates and provides a set of temporal laws to allow both qualitative and quantitative analysis by generating minimal cut sequences (MCSQs) from TFTs. Similar to the MCSs of classical fault trees; the MCSQs of TFTs are the smallest sequences of events that are necessary and sufficient to cause the top event. However, the techniques used for solving classical FTs are not suitable to solve Pandora TFTs. The solution requires generating all possible reachable system states and stochastic transitions between states. In other words, a conversion to a Continuous Time Markov Chain (CTMC) is required to solve TFTs. As stochastic Petri Nets (SPNs) are a well-established modelling technique and their underlying reachability graphs are isomorphic to CTMCs, they have been used for state-space solution to DFTs (Codetta-Raiteri, 2005; Zhang, Miao, Fan and Wang, 2009). However, no attempts have been made so far to use stochastic Petri Nets to solve Pandora TFTs.

One of the advantages of Pandora is that it can do qualitative analysis and create useful insight to system failure in the absence of limited or absent quantitative failure data, e.g. in the case of new software components. In addition, the technique is integrated well in model-based design and analysis. It has been shown by Walker and Papadopoulos (2009) that Pandora logical expressions can be used to describe the local failure behaviour of components and then enable synthesis of TFTs from systems models that have been annotated with Pandora expressions using popular notations, e.g. Matlab Simulink, SysML, EAST-ADL, or AADL. Given the increasing importance of model-based design and analysis, and the potential use of Pandora in this context, we believe that it is both theoretically and practically useful to explore possible ways for improved analysis of Pandora TFTs. Therefore, in this paper, we show how the Petri Nets can also be used to solve Pandora TFTs.

2. PANDORA TEMPORAL FAULT TREES

Pandora defines three temporal gates: Priority-AND (PAND), Priority-OR (POR), and Simultaneous-AND (SAND) to extend classical fault trees. These gates allow analysts to represent sequences or simultaneous occurrence of events, and thus enable fault trees to capture sequence dependent dynamic behaviour as well as combinatorial failure

behaviour. Fault tree symbols for the three gates are shown in Fig.1, where (I) is the Priority-AND, (II) is the Priority-OR, and (III) is the Simultaneous-AND.

PAND gate is not a new gate and has been used in FTA as far back as the 1970s (Fussell, Aber and Rahl, 1976), and also used in the Dynamic Fault Trees. However, behaviour of this gate was never properly defined for use in qualitative analysis, resulting in ambiguous outcome. The symbol '<' is used to represent the PAND gate in logical expressions, i.e., A < B means (A PAND Y) where A and B are both failure events. In Pandora, therefore, the PAND gate is defined as being true only if:

1. All input events occur
2. Input events occur in sequence from left to right
3. No input events occur simultaneously

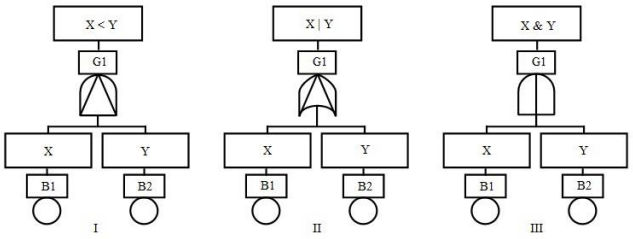

Fig. 1. Temporal gates: (I) PAND. (II) POR. (III) SAND.

Like the PAND gate, the POR gate also defines a sequence, but it specifies an ordered disjunction rather than an ordered conjunction. It is used to indicate that one input event has priority and must occur first for the POR to be true, but does not require all other input events to occur as well. The POR can therefore be used to represent trigger conditions where the occurrence of the priority event means that subsequent events may have no effect. The symbol '|' is used to represent the POR gate in logical expressions, thus A|B means (A POR Y). The POR is true only if the following conditions are true:

1. Its left-most (priority) occurs
2. No other input event occurs before the priority event
3. No other input event occurs at the same time as the priority event

The SAND gate is used to define situations where an outcome is only triggered if two or more events occur approximately simultaneously. For example, this can happen because of a common cause, or because the events have a different effect if they occur approximately simultaneously as opposed to in a sequence. It is true only if:

1. All input events occur
2. All the input events occur at the same time

The symbol '&' is used to represent the SAND gate in logical expressions. In this paper, we use '+' to represent OR and '.' to represent AND gate. In a logical expression the SAND gate has the highest priority, then PAND, POR, AND, and OR. Hence A+B&C<D|E is equivalent to A+(((B&C)<D)|E).

Pandora extends fault trees with temporal gates and provides a set of temporal laws to facilitate qualitative analysis. These laws form the basis for qualitative analysis of Pandora's temporal fault trees and they can all be proved with the help of temporal truth tables as in (Walker, 2009). Temporal laws help to reduce and minimise the failure expressions to obtain minimal cut sequences (MCSQs). From the MCSQs, it is possible to understand what combinations and sequences of events are necessary and sufficient to cause system failure. Pandora considers occurrence of failure events as instant (i.e., go from 'false' to 'true' with no delay) and persistent (i.e., once occurred, they remain in a 'true' state forever).

Although the primary goal of creating Pandora TFTs was to facilitate qualitative analysis, efforts have also been made to enable quantitative analysis. Methodologies for probabilistic evaluation of Boolean gates are available in the Fault Tree Handbook (Vesely et al., 2002). Similarly, as the PAND gate also features in DFTs, algebraic (Fussell, Aber and Rahl, 1976; Merle, Roussel and Lesage, 2011), Markov chain based (Boudali, Crouzen and Stoelinga, 2007), Bayesian Network based (Boudali and Dugan, 2005; Neil et al., 2008; Montani, Portinale, Bobbio and Codetta-Raiteri, 2008), and Petri Net based (Codetta-Raiteri, 2005) methods are all available for quantifying the PAND gate in DFTs. Recently, analytical approaches have been introduced by Edifor, Walker and Gordon (2012, 2013) and a Bayesian Network based approach has been proposed by Kabir, Walker and Papadopoulos (2014) to quantify Pandora temporal fault trees. However, this latter approach models the failure behaviour of the system in a discrete-time domain, and therefore requires settling the granularity of time discretisation as part of the model transformation.

3. PRELIMINARIES ON PETRI NETS

Petri Nets are a formal graphical and mathematical modelling tool widely used with distributed and concurrent systems. Classical Petri Nets consist of a finite set of places, a finite set of transitions, and a finite set of directed arcs (see Fig. 2).

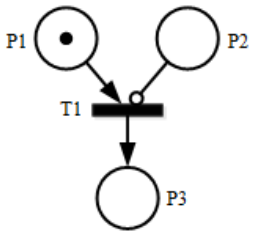

Fig. 2. A simple Petri Net

Places are graphically represented by circles and may contain tokens, while transitions are graphically represented by rectangles and are set to fire when a certain pre-specified number of tokens are available in the places connected to the transitions. Directed arcs connect places to transitions and vice versa. Traditional Petri Nets allow determination of the qualitative properties of systems; if they are extended with temporal data then quantitative and time-dependent analysis can also be performed. Stochastic Petri Nets (SPNs) (Molloy,

1982) are an extension of classical Petri Nets where all transition delays are exponentially distributed. In SPNs, immediate transitions are difficult to model. Generalised Stochastic Petri Nets (GSPNs) (Marsan et al., 1996) solve this by allowing both immediate and timed transitions. Timed transitions fire after a random period of time (defined as exponential distribution of firing time) and immediate transitions fire as soon as they are enabled. Immediate transitions have priority over timed transitions, i.e., if a timed and an immediate transition are enabled at the same time, and if any conflict exists between them, then the immediate transition fires first. In this paper, timed transitions are represented as white rectangles and immediate transitions are represented as black rectangles. A special type of arc known as an *inhibitor arc* that ends with a small circle instead of an arrowhead is used to connect a place to a transition in order to disable the transition if the place is not empty. This allows checking the non-occurrence of events.

Petri Nets can be used to model both nominal and failure behaviour of systems. In some cases, Petri Nets are used to model functional behaviour and then another safety analysis method e.g., FTA is used to analyse failure behaviour of the systems based on the non-functional behaviour identified from the Petri Nets. In order to increase the modelling and analytical capability of combinatorial approaches to dependability analysis, Bobbio, *et al* (1999) and Helmer *et al*. (2007) have proposed ways of translating fault trees into PNs. Codetta-Raiteri (2005) and Zhang *et al* (2009) have introduced ways of mapping DFTs to PNs.

4. BEHAVIOURAL SPECIFICATION OF TFT GATES WITH PETRI NETS

This section defines a method for translating Pandora temporal fault trees into generalised stochastic Petri Nets. In Pandora TFT, events are considered as non-repairable, i.e., once a component fails it stays in the failure state forever, and gates propagate faults instantly. In the proposed mapping, each TFT node (basic, intermediate, and top events) is mapped to a sub-net where there is a place indicating the status of the node. We use places to represent the state of the system, timed transitions to represent random faults and immediate transitions to represent failure propagation. Timed transitions are characterised by the failure rate of basic events and here we have used exponential distribution of failure rates. The mapping of each TFT gate to GSPN should be correct in that there is a place in the sub-net representing the outcome of the gate and if all the conditions are fulfilled for the transition representing a gate to fire then the place gets a token.

4.1 Basic Events

The mapping of a basic event to a Petri Net is shown in Fig.3. As all components are assumed to be fully functional at time 0, the place representing the working state has a token. Every component has a failure rate, and the occurrence of an event (component failure) is represented by the timed transition named *Fail.* Once this transition fires, the event occurs and a token is consumed from the place *Working* and a token is deposited to the place *Failed* which represents a failure state of a component. After that the failure is propagated instantly through the immediate transition named *Propagate.* The outgoing arrow from transition *Propagate* back to place *Failed* serves to maintain persistency of events, i.e., maintain the permanent failure state of the event irrespective of any further propagation. To ensure that the error is propagated exactly once in a single path, the inhibitor arc from place *Propagated* to transition *Propagate* is used.

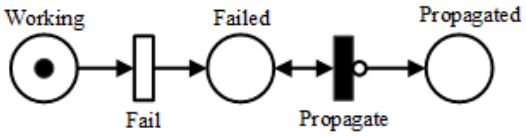

Fig. 3. Mapping of a basic event to a Petri Net

4.2 Temporal Fault Tree Gates

Mapping of Boolean AND and OR gates to a Petri Net is done based on the work of Bobbio *et al*. (1999) and shown in Fig.4 and 5 respectively.

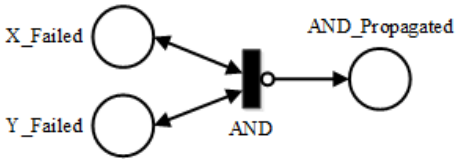

Fig. 4. Mapping of a two input AND gate to a Petri Net

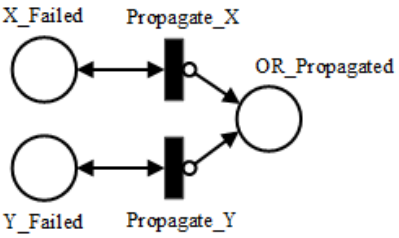

Fig. 5. Mapping of a two input OR gate to a Petri Net

All the places corresponding to the input events (X_Failed, Y_Failed) of the AND gate are connected to a single immediate transition with bidirectional arrows. The transition will fire when all the input places have tokens (i.e., all input events occur) and on firing it deposits a token to the place representing the outcome of the gate (*AND_Propagated*). The OR gate is translated to PN by creating a transition for each place corresponding to input events of the gate. When one of the input places gets a token, one of the transitions will fire and deposit a token to the place corresponding to the outcome of the OR gate (*OR_Propagated*). In both the cases (AND and OR) no token is consumed from the input places because an event may be the input of several gates (hence the bi-directional arrows). If there are more inputs either to the AND or the OR gate then we just need to include them following the same fashion shown in Fig. 4 and 5.

Boolean gates are stateless in a sense that they do not need to remember the order of occurrence of the events. However, temporal gates must remember the order of occurrence of input events. The transformation of a two input PAND gate is shown in Fig. 6. If there is a token in *X_Failed* and no token in *Y_Failed* then the transition *Propagate_X* will fire and deposit a token to *X_NOT_Y* , i.e., event X has happened but Y has not happened yet. Afterwards, if *Y_Failed* gets a token due to the occurrence of event Y then this in conjunction with *X_NOT_Y* will enable the transition named *PAND* to fire, and thus deposit a token to the place corresponding to the outcome of the PAND gate (*PAND_Propagated*). A PAND gate with more input can be mapped to a Petri Net by following this technique whilst maintaining the strict sequencing of events.

Fig.7 shows the mapping of a two input POR gate to a Petri Net. There is a priority event in the POR gate and the logic of the POR gate dictates that to make the POR output true only the priority event is required to occur, or if other events also occur then they should occur after the priority event. In the POR gate of Fig.7, event *X_failed* has priority over event *Y_Failed*. The place *POR_Propagated* represents the scenario when the priority input event occurs first and if any other input occur then they occur after the priority event. In case of more than two inputs we have to make sure that either none of the non-priority input occurs or disjunction of all the non-priority events occurs after the priority event to make the POR outcome true.

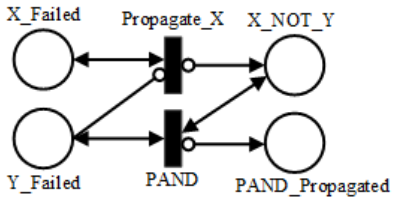

Fig. 6. Mapping of a two input PAND gate to a Petri Net

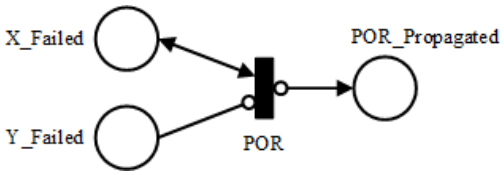

Fig. 7. Mapping of a two input POR gate to a Petri Net

As we use the exponential firing rates (failure rates) for the timed transitions, two transitions firing at the same time is zero. For this reason, if we transform the SAND gate containing two or more basic events into a Petri Net then during the whole mission time we will not get any token in the place representing the SAND output, i.e., probability of the SAND outcome is always 0. For this reason, during quantitative analysis using exponential failure rates we can ignore any MCSQ that contains a SAND gate and no need to transform a SAND gate to PN. The logical correctness of all the mappings of TFT gates to Petri Nets was verified by testing this scheme in the CPN tool (Jensen, Kristensen and Wells, 2007).

5. CASE STUDY AND EVALUATION

We have applied the proposed approach to a case study of a simplified fault tolerant fuel distribution system of a ship, originally used in (Edifor, Walker and Gordon, 2012), reworked and shown in Fig. 9. In the functional mode of the system, there are two primary fuel flows: Tank 1 provides fuel to Engine 1 through Pump 1 (P1), and Tank 2 provides fuel to Engine 2 through Pump 2 (P2). Flowmeter 1 (F1) and Flowmeter 2 (F2) observe the rate of fuel flow to Engine 1 and Engine 2 respectively and provide observed information to the controller. On detecting discrepancy in the fuel flow to either engine, the Controller introduces dynamic behaviour to this system to handle the situation by activating the standby Pump 3 (P3), redirecting fuel flow accordingly by opening some of the valves from V1-V4.

For example, if inadequate fuel flow to Engine 1 is detected, then the Controller can open Valve 1 and 3 (V1 and V3) and activate Pump 3 (replacing Pump 1), and thus facilitate fuel flows to Engine 1 through Pump 3 instead of Pump 1. In contrast, if insufficient fuel flow to Engine 2 is detected then Pump 3 will be activated and Valve 2 and 4 will be opened instead of Valve 1 and 3. Hence, Pump 3 can replace either Pump 1 or Pump 2, but not both. A failure of Pump 1 and Pump 2 will result in no fuel flow to at least one engine; e.g., if Pump 2 fails and Pump 3 replaces it, then Pump 3 will be unavailable for replacing Pump 1 if the latter fails. This results in degraded propulsion functionality for the ship and with one engine working only the speed and manoeuvrability of the ship will be reduced.

Pandora temporal gates can be used to capture the dynamic behaviour of the fuel distribution system and correctly capture the sequence of events together with the combinations of the events that can cause system failure. For simplicity, the internal failure of the engines themselves is left out of the scope of the analysis. The minimal cut sequences for the failure behaviour of Engine 1 of the fault tolerant fuel distribution system was obtained via model-based synthesis from Pandora descriptions of local failure logic of components, and the minimal cut sequences are:

```
E1 = (P1|P2).P3 + (P1|P2).V1 +(P1|P2).V3
   + (S1<P1)|P2 + (S1&P1)|P2 + P1&P2
   + (CF<P1)|P2 + (CF&P1)|P2 + P2<P1
```

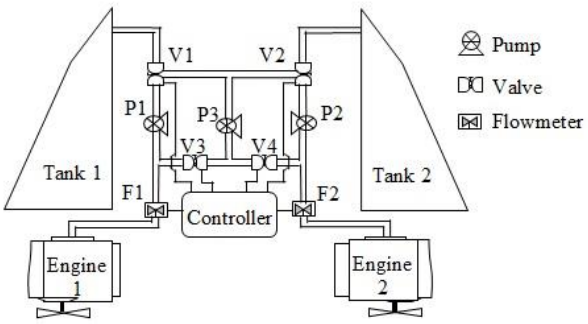

Fig. 8. Fault Tolerant fuel Distribution System

The basic events in the MCSQs are:

1. P1/P2/P3 = Failure of Pump 1/2/3
2. V1/V3 = Failure of Valve 1/3
3. S1 = Failure of Flowmeter 1
4. CF = Failure of Controller

As failure of Engine 1 and Engine 2 are caused by the same events in the opposite sequence, the analysis of failure behavior of Engine 2 is omitted for brevity. In the continuous time domain, the probability of two exponentially distributed independent events occurring exactly at the same time is effectively 0, therefore probability of the MCSQs containing SAND gate (S1&P1|P2, CF&P1|P2, and P1&P2) are considered as 0, and thus the equivalent Petri Net models for those MCSQs are not created. Note that this assumption may not always be true, e.g., in cases where a sizable mass of probability may exist for simultaneity. The equivalent Petri Net model for all other MCSQ are created following the procedure described in Section 4 and the Petri Net model for the failure behaviour of Engine 1 is obtained by combining the Petri Net model of all MCSQ, and shown in Fig. 9. In this figure, timed transitions (white rectangles) are characterised by exponential firing rates based on the failure rate of the components they are connected with. All the components are thought to have a constant failure rate per hour with probability of occurrence following an exponential distribution and the values are shown in Table 1.

The system unreliability was calculated for mission times ranging from 2000 hours to 20000 hours using the ORIS tool (Horváth *et al*, 2012). The system unreliability for different mission times is shown in Table 2. For comparison, considering mission time as 10000 hours, the Bayesian Network based technique (Kabir, Walker and Papadopoulos, 2014) for quantifying the TFT yields the unreliability of the system as 0.1159 and 0.1187 with 4 and 5 time slots respectively. For mission time as 10000 hours, the unreliability value obtained by the Petri Net based technique proposed in this paper is 0.1170 and it is quite close to the above mentioned values.

Table 1. Failure rates of components of fault tolerant fuel distribution system

Component	Failure rate/hour (λ)
Valve 1	1.0E-5
Valve 3	6.0E-6
Pump1 & Pump2 & Pump3	3.2E-5
Flowmeter	2.5E-6
Controller	5.0E-7

Table 2. Unreliability of fuel distribution system

Mission Time (Hours)	Unreliability
2000	0.007
4000	0.024
6000	0.050
8000	0.081
10000	0.117
12000	0.155
14000	0.195
16000	0.235
18000	0.275
20000	0.315

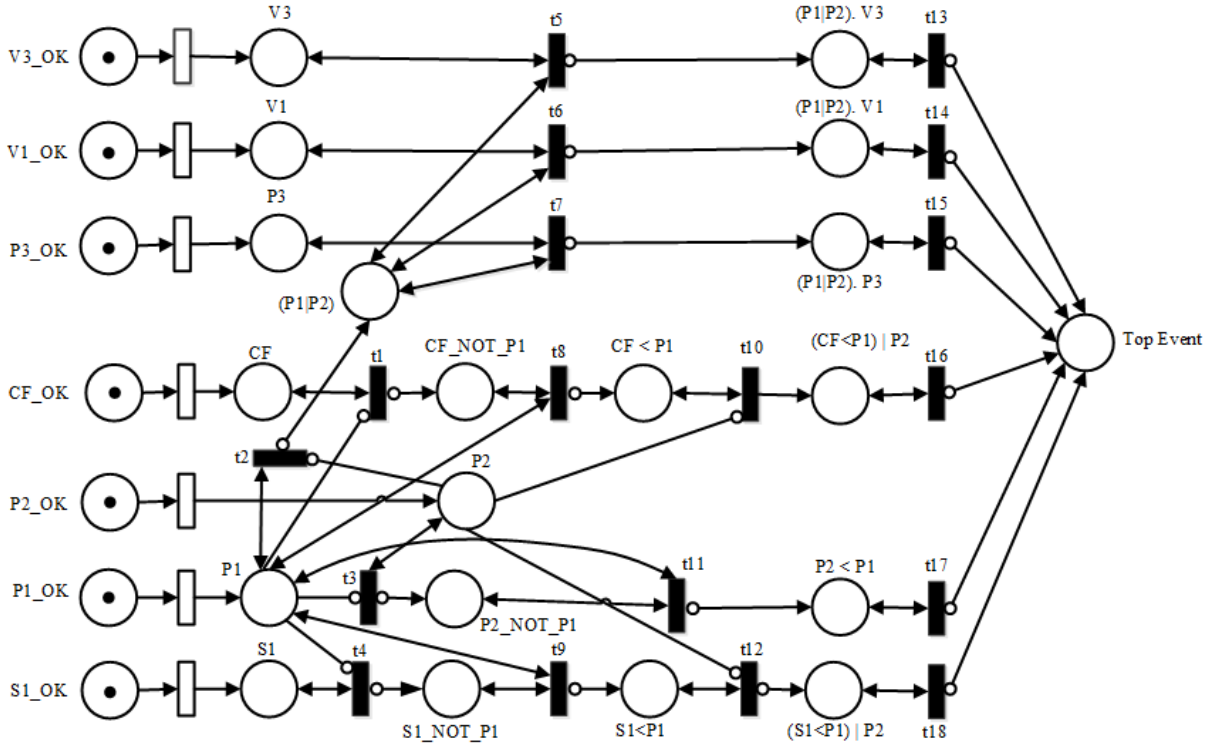

Fig. 9. Petri Net model of the failure behaviour of the fuel distribution system

6. CONCLUSION

In this paper, we have described how Pandora TFTs can be used to perform dynamic dependability analysis. Pandora is easily integrated in popular model-based design and analysis techniques, but classical combinatorial FTA approaches cannot be used for the quantitative analysis of Pandora. In this paper we show how Petri Nets provide a state space solution to Pandora TFTs and presented a method for transforming TFTs to Petri Nets for the purpose of dynamic dependability analysis. After creating a Petri Net for an example fuel system, the system unreliability was calculated for different mission times. One difficulty we foresee is that state space analysis could be computationally expensive since the number of states increases exponentially with the number of components of the system. Therefore, instead of analysing the whole TFT using state-based techniques, in the future we plan to use modularisation techniques that will enable us to use combinatorial solutions for modules with Boolean gates and state space solutions for modules with temporal gates.

REFERENCES

Bobbio, A., Franceschinis, G., Gaeta, R. and Portinale, L. (1999). Exploiting Petri Nets to Support Fault Tree Based Dependability Analysis. In: *8th International Workshops on Petri Nets and Performance Models*. Zaragoza: IEEE, pp.146–155.

Boudali, H., Crouzen, P. and Stoelinga, M. (2007). Dynamic Fault Tree analysis using Input / Output Interactive Markov Chains. In: *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Washington DC: IEEE Computer Society, pp.708–717.

Boudali, H. and Dugan, J.B. (2005). A new bayesian network approach to solve dynamic fault trees. In: *Proceedings of Annual Reliability and Maintainability Symposium*. IEEE, pp.451–456.

Codetta-Raiteri, D. (2005). The Conversion of Dynamic Fault Trees to Stochastic Petri Nets, as a case of Graph Transformation. *Electronic Notes in Theoretical Computer Science*, 127(2), pp.45–60.

Dugan, J.B., Bavuso, S.J. and Boyd, M.A. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3), pp.363–377.

Edifor, E., Walker, M. and Gordon, N. (2012). Quantification of Priority-OR Gates in Temporal Fault Trees. In *Computer Safety, Reliability, and Security SE - 9*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp.99–110.

Edifor, E., Walker, M. and Gordon, N. (2013). Quantification of Simultaneous-AND Gates in Temporal Fault Trees. In: *New Results in Dependability and Computer Systems SE - 13*, Advances in Intelligent Systems and Computing. Springer International Publishing, pp.141–151.

Fussell, J.B., Aber, E.F. and Rahl, R.G. (1976). On the Quantitative Analysis of Priority-AND Failure Logic. *IEEE Transactions on Reliability*, R-25(5), pp.324–326.

Helmer, G., Wong, J., Slagell, M., Honavar, V., Miller, L., Wang, Y., Wang, X. and Stakhanova, N. (2007). Software fault tree and coloured Petri net – based specification , design and implementation of agent-based intrusion detection systems. *International Journal of Information and Computer Security*, 1(1), pp.109–142.

Horváth, A., Paolieri, M., Ridi, L. and Vicario, E. (2012). Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation*, 69(7-8), pp.315–335.

Jensen, K., Kristensen, L.M. and Wells, L. (2007). Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4), pp.213–254.

Kabir, S., Walker, M. and Papadopoulos, Y. (2014). Reliability Analysis of Dynamic Systems by Translating Temporal Fault Trees into Bayesian Networks. In: *Model-Based Safety and Assessment*, Lecture Notes in Computer Science. Cham: Springer International Publishing, pp.96–109.

Marsan, M.A., Balbo, G., Conte, G., Donatelli, S. and Franceschinis, G. (1996). *Modeling With Generalized Stochastic Petri Nets*. West Sussex: Wiley.

Merle, G., Roussel, J.-M. and Lesage, J.-J. (2011). Algebraic determination of the structure function of Dynamic Fault Trees. *Reliability Engineering & System Safety*, 96(2), pp.267–277.

Molloy, M.K. (1982). Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, c-31(9), pp.913–917.

Montani, S., Portinale, L., Bobbio, A. and Codetta-Raiteri, D. (2008). Radyban: A tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks. *Reliability Engineering & System Safety*, 93(7), pp.922–932.

Neil, M., Tailor, M., Marquez, D., Fenton, N. and Hearty, P. (2008). Modelling dependable systems using hybrid Bayesian networks. *Reliability Engineering & System Safety*, 93(7), pp.933–939.

Vesely, W., Stamatelatos, M., Dugan, J., Fragola, J., Minarick, J. and Railsback, J. (2002). *Fault tree handbook with aerospace applications*. NASA office of safety and mission assurance, Washington DC.

Walker, M. (2009). *Pandora: A Logic for the Qualitative Analysis of Temporal Fault Trees*. PhD Thesis, University of Hull.

Walker, M. and Papadopoulos, Y. (2009). Qualitative temporal analysis: Towards a full implementation of the Fault Tree Handbook. *Control Engineering Practice*, 17(10), pp.1115–1125.

Zhang, X., Miao, Q., Fan, X. and Wang, D. (2009). Dynamic fault tree analysis based on Petri nets. In: *8th International Conference on Reliability, Maintainability and Safety(ICRMS)*. Chengdu: IEEE, pp.138–142.