# Phased Mission Modelling of Systems with Maintenance Free Operating Periods Using Simulated Petri Nets

**S. P. Chew, S. J. Dunnett and J. D. Andrews**
Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, Leics, UK

## Abstract

A common scenario in engineering is that of a system which operates throughout several sequential and distinct periods of time, during which the modes and consequences of failure differ from one another. This type of operation is known as a phased mission, and for the mission to be a success the system must successfully operate throughout all of the phases. Examples include a rocket launch and an aeroplane flight. Component or sub-system failures may occur at any time during the mission, yet not affect the system performance until the phase in which their condition is critical. This may mean that the transition from one phase to the next is a critical event that leads to phase and mission failure, with the root cause being a component failure in a previous phase. A series of phased missions with no maintenance may be considered as a Maintenance Free Operating Period (MFOP). This paper describes the use of a Petri net to model the reliability of the MFOP and phased missions scenario. The model uses a form of Monte-Carlo simulation to obtain its results, and due to the modelling power of Petri Nets, can consider complexities such as multi-mission periods, component failure rate interdependencies, and mission abandonment. The model operates three different types of Petri Net which interact to provide the overall system reliability modelling.

**Keywords:** phased missions, Petri nets, maintenance free operating period, MFOP

## 1. Introduction

The success of a mission may depend upon the completion of a sequential series of objectives of varying time intervals. If this is the case, the mission may be referred to as a *phased mission*, with each individual time period referred to as a *phase*. The phases of a mission may be distinguished by phase number, time interval, system configuration, desired tasks, performance metrics, etc, and may differ from one another such that the logic, modes and consequences of system failure are different. An example of this type of mission is a military aircraft flight pattern, with phases such as taxi to runway, take-off, ascent, level flight to target, ingress, attack, egress, level flight to runway, descent, land, and taxi to base. Combining several sequential phased missions without maintenance may be considered to produce a Maintenance Free Operating Period (MFOP). These are discussed further in Section 2.

There is a need to express the phase and mission failures in terms of the various system, sub-system or component level (basic event) failures that can cause them. It is also necessary to be able to quantify the top event occurrence probability and frequency from the reliability information of the basic events.

The main techniques used in solving phased mission problems are fault tree analysis (FTA), Markov analysis and simulation. FTA is a widely-used method of assessing the failure probability of non-repairable systems, representing the causes of a particular failure in terms of basic events (such as component failures). This has been extended to phased missions where each phase has a different failure logic model, making the modelling of the scenario more complex. The first publication of work regarding phased missions and fault trees was written by Esary and Ziehms [1]. They considered the analysis of non-repairable systems, regarding the success of a mission as being the successful completion of all the phases therein. Conversely, the failure of a mission is expressed as the loss of function of the system during at least one of the phases. The probability of this is the mission unreliability. The paper shows how the phase fault trees can be combined into one overall mission fault tree. An effective solution method is given for mission but not phase unreliability. The key problem is the calculation, as efficiently as possible, of either the exact value or the bounds of the mission unreliability metric. Burdick *et al* [2] took this method further, making the computation of the overall mission fault tree more efficient by allowing the exclusion or combination of certain cut sets, and discussed methods of estimating bounds to the mission unreliability.

A problem in these earlier works is the lack of ability to compute the probability of system failure in each phase. This may be necessary or useful where the consequences of phase failure differ. Andrews and La Band [3], using non-coherent fault trees (NOT gates included), developed a method of establishing this. They combined the causes of system failure by the end of phase *p* with the causes of system success by the end of phase *p*-1. The Binary Decision Diagram method was used to speed up both the qualitative and the quantitative analysis.

If the independence between component failures cannot be ensured, or if the system being analysed is repairable, then it may be necessary to use a Markov approach to find mission unreliability [4]. The Markov method considers each possible state of the system and the transitions between these states in terms of component failures or repairs. Transition rates correspond to the failure and repair rates of components from which the probability of phase success and failure can be established. By considering the conditions at the end of one phase as the initial conditions of the next phase, and reapplying the method throughout the mission, a value of mission reliability can be reached. It is not possible to find the mission reliability by simply multiplying the phase reliabilities, due to the statistically dependent nature of the phases. At phase change times, the system must occupy a state allowing both phases to function, to progress to the next phase or the mission will fail at these times. Smotherman and Zemoudeh [5] consider a non-homogeneous Markov model (where the component failure rates change with time), and generalise state transitions to include phase changes as well as component failure. They solve this model using a numerical solution, an adapted fifth-order Runge-Kutta method. In a later paper, Smotherman and Geist [6] describe a similar approach but include a reward model to provide figures of merit for work performed.

Simulation techniques can also be used to model phased missions, as their computational nature allows the inclusion of many complexities of analysis which cannot be considered with techniques such as Fault Tree Analysis or Markov modelling. This includes using a broad range of different component failure time distributions and repair queuing, as well as all the benefits of the other methods, such as component failure rate interdependence and repairable basic events. [7] discusses phased missions and simulation.

One method that allows simple graphical representation as well as significant modelling power is the Petri net. Petri nets are explained in Section 3. They can be applied to the field of phased missions, and allow for the inclusion of many different system designs.

This paper discusses the analysis of Phased Missions using Petri nets, and provides a method of modelling Maintenance Free Operating Periods. Discussed in Section 2, this is a figure describing the usefulness of a system with regards to carrying out military missions. To date, no papers have been published discussing the combined analysis of Phased Missions and MFOP.

## 2. Maintenance Free Operating Period

The concept of the Maintenance Free Operating Period, or MFOP, was first proposed in 1996 by the UK Ministry of Defence as a means of aiding manufacturers of military aircraft to meet its needs [8]. These needs include better operational planning capability, improved operational availability of aircraft and reduced running costs.

The MFOP is described as a period of operation during which the equipment must be able to carry out all its assigned missions without any maintenance action and without the operator being restricted in any way due to system faults or limitations [9]. Attached to this idea is that of MFOP Survivability, which is the confidence level of successfully completing the MFOP. Following each MFOP is a period, known as a Maintenance Recovery Period or MRP, where the aircraft is repaired to such a level that it is capable of completing the next MFOP. This may not necessarily involve repairing all systems, rather those that are necessary for the completion of all the forthcoming missions.

Five different areas have been identified as being important to achieving a high value of MFOP with a high confidence level [10]:

1. Inherent reliability of systems and components: By improving the quality and reliability levels of these, and understanding better the reasons for their failure and what can be done to prevent them, an increase in the reliability of the platform will result.

2. Redundant Systems: MFOP has not been designed to provide complete failure-free operation throughout the period; rather, it involves the understanding that, upon the failure of a system or component, the platform should be able to withstand this and continue its operation. Redundant systems provide the most obvious way of doing this. They will usually remain dormant until the failure of a system, at which point they are brought online to provide the same functions as the now failed system.

3. Reconfigurable Systems: Another way of allowing the platform to continue normal operation after the failure of a system is by providing reconfigurable systems. These are usually online during a mission, but with the ability to alter their behaviour to take into account the failure of a system.

4. Prognostics: The ability to detect the likely failure of a component or system within the next MFOP would be of great benefit. Health Monitoring systems, greater understanding of the life expectancy and reasons for failures of systems and components, and better inspection methods will allow a system to be replaced before an MFOP, with confidence that it would not have lasted to the next MRP.

5. Diagnostics: The location and isolation of a particular failed component or system will enable the reconfiguration of systems or mission objectives.

There is currently a tool available which is capable of analysing these aspects of MFOP, known as the Ultra-Reliable Aircraft Model (URAM) [11, 12]. A bespoke discrete event simulation tool, it is capable of considering many aspects of real-life reliability analysis, such as various reliability distributions, environmental considerations, usage wear and many aspects of maintenance (repair queuing, spares management, etc.). The approach presented in this paper is designed to extend the modelling capability currently provided by URAM.

## 3. Petri Nets

First created in 1962 and reported in the thesis of C.A. Petri [13], Petri Nets (PNs) are an adaptable and versatile, yet simple, graphical modelling tool used for dynamic system representation. The various modelling applications of PNs to date include use in computer software and hardware systems, manufacturing systems and reliability evaluation. They have undergone much adaptation and variation from the initially proposed diagrams, and, often confusingly, many of these variants exist concurrently, which could be a factor hindering their wide adoption throughout industry.

A Petri Net is a bipartite directed graph with two types of node: *places*, which are circular, and *transitions*, shown as bars. Places link only to transitions, and vice versa, using directed arcs, and each may have infinite inputs/outputs. It is possible for a place to have several arcs to or from the same transition, which is condensed down into a single arc with a *weight* or *multiplicity*, and denoted by a slash through the arc with a number next to it [14]. If there is no slash, the multiplicity is assumed to be 1. The dynamic aspect of the Petri Net is formed by *tokens* or *marks*, which abide within places, and are passed between them by the *switching* of transitions. An example of transition switching is shown in Figure 1.

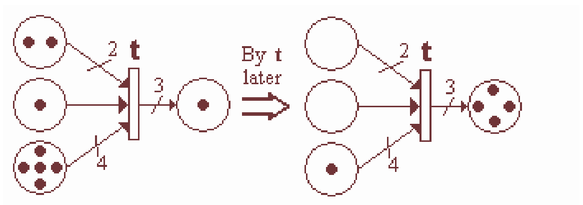


Figure 1 – Transition Enabling and Switching

The figure shows a transition which has three places as inputs. The first place has an arc weight of 2, the middle one has a single multiplicity while the third has a weight of 4. Because the number of tokens in all of the input places to the transition contains at least the weight-number of tokens, that transition can be said to be *enabled*.

Although the original Petri Nets [13] did not have a concept of time, transitions can also be associated with a delay, which forces the transition to postpone switching for a period upon being enabled [15]. This delay can be zero (in which case the transition is drawn as a solid bar), deterministic or randomly sampled from a given distribution[16-18]. In Figure 1, there is a time delay of **t** applied once the transition is enabled.

Finally, once the time period has passed and the transition remains enabled, the switching takes place. This process removes and 'destroys' the number of tokens in each input place corresponding to the multiplicity of the relevant arc, and 'creates' the weight-number of tokens in each output place. This is shown in Figure 1 where the switching removes 2, 1 and 4 tokens from each of the input places, and deposits three tokens in the output place. The transition is then disabled, as the input places do not have the correct number of tokens.

It is possible to prevent a transition switching by using an *inhibitor arc*. This special arc, shown by a line with a small circle on the end instead of an arrow, connects only an input place to a transition: see Figure 2. It acts such that if the number of tokens within the place is at least that of the arc weighting, the transition cannot switch, regardless of whether it is enabled or not [19]. In Figure 2, the otherwise enabled transition can wait for time **t** to expire, but cannot switch – no tokens are moved by that transition while the inhibiting place contains the relevant number of tokens.

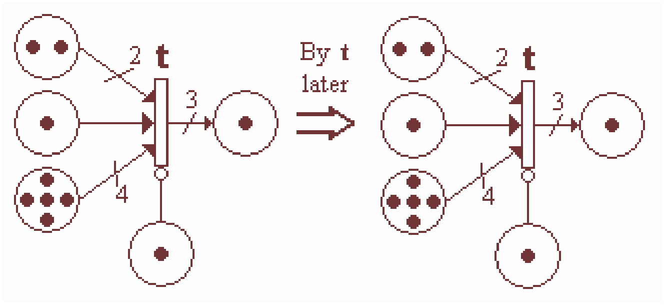


Figure 2 – Inhibitor arc preventing switching

It is the switching of the transitions which represents the dynamic behaviour of the Petri net model – the ability to transport tokens around the net, thereby changing the *marking* with each switch. The net marking is a term given to the distribution of the tokens throughout the whole Petri net, and each form of it represents a different system state. It is this which is of interest to the analyst.

The following section details a new method of modelling phased missions and MFOPs, using three distinct types of PNs.

## 4. Modelling Phased Missions

Mura and Bondavalli [20], in their paper detailing a method of modelling phased missions using Petri nets, employed a concept of having two distinct Petri nets to model the situation. The first was the Phase Net, which showed the progression from the first phase of the mission to the last. It allowed for the mission profile to include changes of mission, for instance where a primary objective has to be abandoned in favour of a secondary objective, thus making the method more relevant. The current phase then fed into and affected a separate PN known as the System Net, which governed the system failure throughout the mission.

The method proposed in this paper to model phased missions extends this approach, allowing analysis of a more complex scenario, and offers a more structured modelling technique, using three different types of net:

- *Phase Petri Nets* (*PPN*) – Each phase of the mission has a given failure logic which expresses the system failure in that phase in terms of component or basic event failures. This logic is expressed here in Petri net form.
- *Component Petri Net* (*CPN*) – Fails components according to randomly sampled times and allows their repair at the end of missions
- *Master Petri Net* (*MPN*) – Governs phase progression, mission abandonment and entering period of maintenance for components.

The different Petri nets must interact, and that function is provided by arcs linking places and transitions in the relevant Petri nets. The following sections consider these elements of the proposed model in more detail:

### 4.1. *Phase Petri nets*

The PPNs express the system failure logic in a particular phase, in terms of the basic events or component failures. In order to develop the PPNs they must be able to model the logic gates which combine the basic event occurrences in the correct way. In standard Fault Tree logic, the two main fundamental logic operators are AND and OR gates, shown along with their Petri net representations in Figures 3 and 4. A transition has AND logic built in, so it is simple to model its behaviour – all input places must have a token to switch the 'gate'. An OR gate has one place and one transition for each input to the gate. If any one of these inputs gets a token, the corresponding transition will switch, depositing it in the output place.

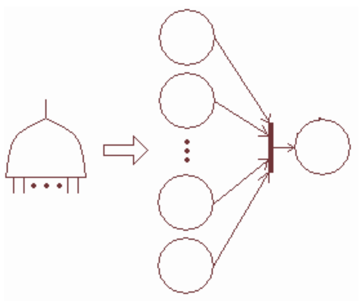


Figure 3 – Petri Net AND Gate

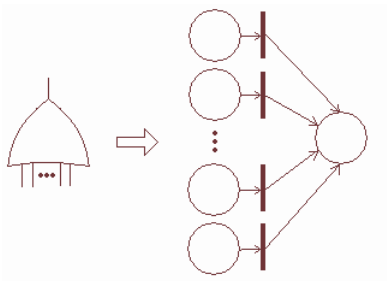


Figure 4 – Petri Net OR Gate

It is also possible to model NOT gates. These are used in non-coherent fault trees and have been used in the modelling of Phased Missions [3]. In the Petri Net for a NOT gate, shown in Figure 5, two transitions are required to model the correct behaviour. In the Figure, where an arc is two-way, that arc is combined into a single arrow, including where an arc inhibits in one direction but is normal in the other. The right-hand transition places a single token in the output place (top) only if there is no token in the input place (bottom left). Once a token exists in the input place, the output place becomes unmarked by the left-hand transition, and no more tokens can enter it until the input place becomes unmarked.

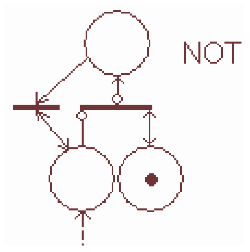
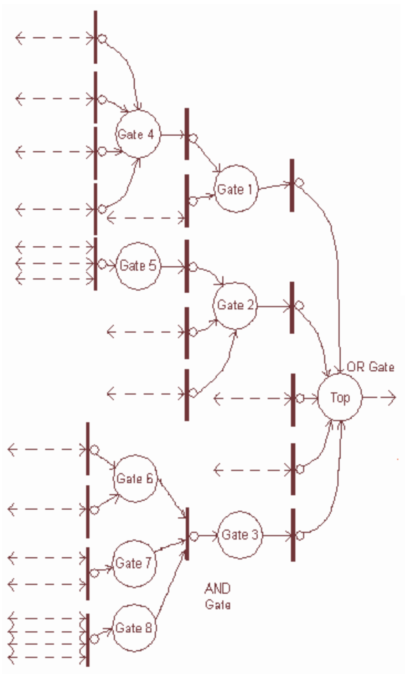


Figure 5 – Petri Net NOT Gate

The PPNs take inputs from the components in the CPN that feed into immediate transitions. These then combine the various basic event inputs into cascading higher level events, using AND, OR, NOT or possibly other logical permutations, with each output place of a gate referring to the occurrence of an intermediate failure event (this could be a more serious occurrence than a basic event or the failure of a sub-system or system). The top event is failure of the overall system or platform in that particular phase. Figure 6(a) shows the general layout of an example PPN, while Figure 6(b) shows the equivalent Fault Tree. The place corresponding to “TOP” has five input transitions, making it a 5-input OR Gate. Of these, two take their tokens directly from component failures provided by the CPN, indicated by the dotted arrow in Figure 6(a), while three come from intermediate events. Inputs to Gate 1 are linked by OR logic; these are Gate 4 and a component failure. Similarly, Gate 2 is an OR gate, with Gate 5 and two component inputs. Gates 5 and 3 are AND gates with three inputs each. Note that the presence of inhibiting arcs from a place to its input transition prevent an infinite number of tokens being passed to it.



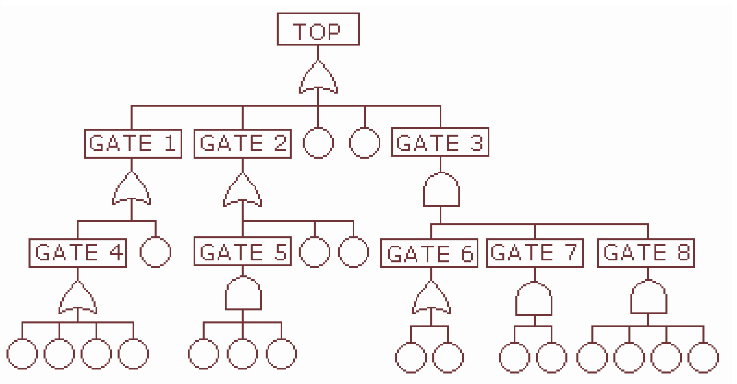
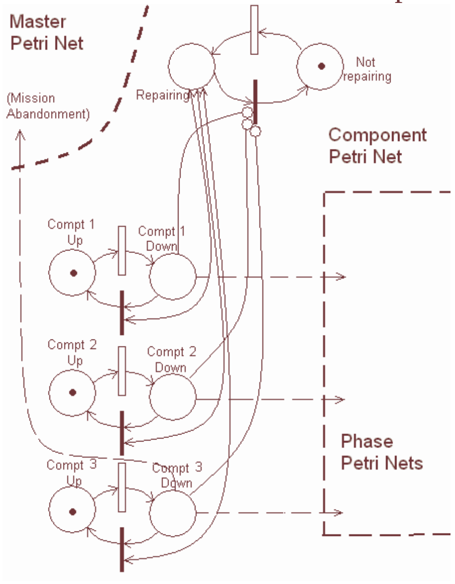


Figure 6(a) – Example PPN Figure 6(b) – Example Phase Fault Tree

### *4.2. Component Petri net*

The Petri net models of basic event failures, which may combine to cause top event failure, are located in the Component Petri Net (CPN). This net also models the repair of the basic events during the MRP.

A basic event is very simply modelled using Petri Net depiction, using two places to define a ‘working’ and ‘failed’ state, and two or more transitions to model the shift between these states. An example CPN is shown in Figure 7.



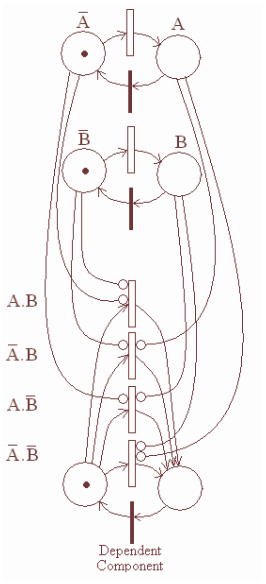


Figure 7 – Example Component PN

Figure 8 – Component Dependencies

If component *n* fails, the token currently in the “compt *n* up” place is passed to “compt *n* down”, which then feeds into a PPN. The model is designed to consider repair only after all missions in a particular sequential set (this can be considered to be a Maintenance Free Operating Period or MFOP) have been completed. When this period of repair, called a Maintenance Recovery Period or MRP, is entered, a token is placed in the “repairing” place at the top of Figure 7. Each component which has failed will find that its immediate transitions are now enabled, and these switch, placing a token back in the “compt *n* up” place and the repairing place. Once all components have been repaired, the immediate end-of-repair transition switches.

#### *4.2.1. Dependency Modelling*

It is conceivable that dependencies can exist in the system and failures do not occur independently. In this circumstance a component failure probability may change depending upon the functionality or failure of a different component or system. A simple example of this is the processor and fan in a computer – if the fan fails, the processor overheats and fails at a higher rate. This behaviour is dealt with in the model by allowing a component to have more than one failure transition. Only one of these can be enabled at any one time, depending on the components or systems which cause the acceleration of failure, as shown in Figure 8. In the figure, the dependent component has four failure transitions, due to the dependency between itself and components A and B. Figure 8 shows the dependent and independent components to be operational. This state means that transitions $A.B$, $\overline{A}.B$ and $A.\overline{B}$ are inhibited, while transition $\overline{A}.\overline{B}$ is operational. If, however, A fails but B remains operational, then transition $A\overline{B}$ would be enabled, while all the others, including $A.B$, would be inhibited. If a component fails, and causes the failure rate of the operational component to increase, it can be regarded that the operational component is now more ‘stressed’. The nature of this stress may vary: it may be due to heat, pressure, humidity, vibration. Whatever the cause, the new time to failure can now be sampled from an alternative distribution.

### *4.3. Master Petri Net*

The net which controls the operation of the simulations and governs the performance of phases and missions is called the Master Petri Net. This is a complex net and as such is considered in three interdependent sections, as shown in Figure 9:

- Control of the sequence of phases, and failure or success of each mission (solid line border)
- Ending each mission or MFOP and performing repairs (dotted line border)
- Abandoning the mission due to specific component or system failures (dashed line border).

The section surrounded by a solid line controls the changing of the phases. Each phase has a place which, if marked, indicates that that phase is currently in operation. These output to timed transitions whose switching times are the lengths of the phases. Whilst these are often considered to be deterministic, it is possible to have randomly sampled phase lengths.

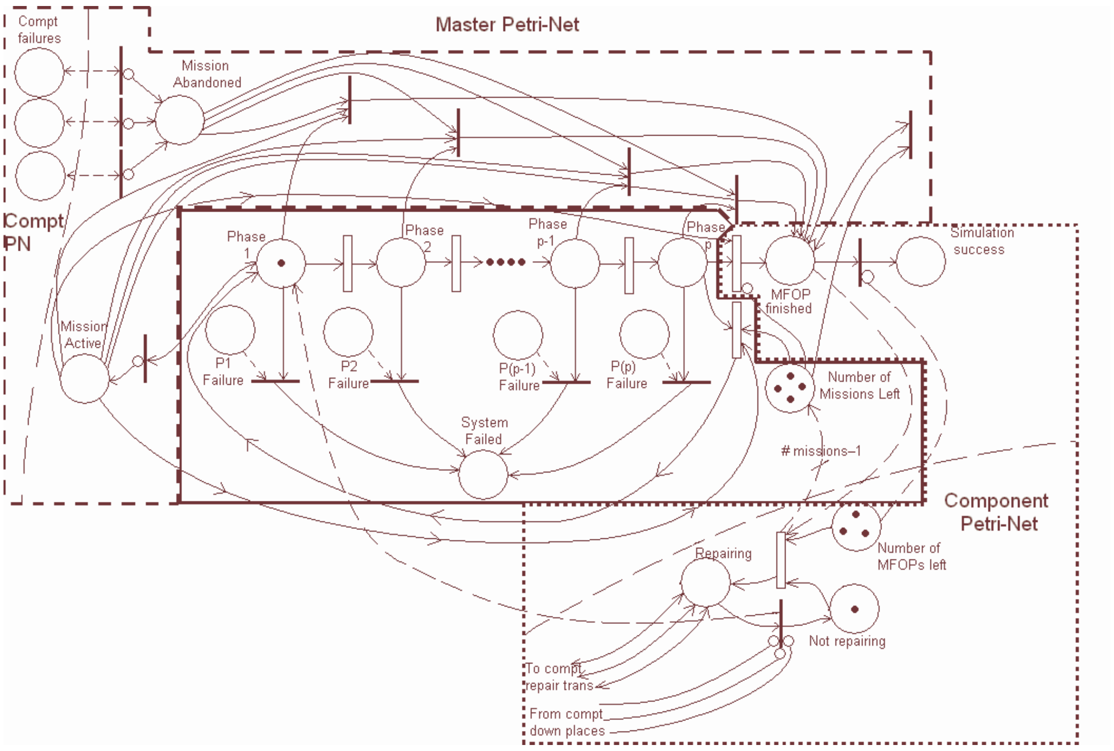


Figure 6 – Master Petri Net

If a particular phase top event occurs, then a token will be present in its “Phase *n* failure” place. Only when a phase has failed and the mission is currently in that phase can the system and mission be considered a failure. This means that if a phase has completed before the top event occurs, then the mission will still complete, whereas if a phase failure occurs before that phase has started, the mission will fail upon phase commencement, and a token will be placed in the “System Failed” place. This style of modelling phases was adapted from [19].

If the mission manages to complete successfully, it is possible either that another mission will take place straight away, before any maintenance, or that the system will enter a period of maintenance. If the former is true, the system will replace the phases token back in the phase 1 place and restart the mission. If not, the token enters the “repair” section of the Master PN. This is the dotted part of Figure 9. The phases token will enter the “MFOP Finished” place to begin system repair, as mentioned in the previous section. Once repair has finished, a new set of maintenance-free missions can begin. Only after all of these have completed and a token is placed in the “Simulation success” place, or a mission failure occurs, is the simulation considered to be over.

The final section has a dashed border in the figure and allows for a mission to be abandoned. This abandonment occurs when a particular basic or intermediate event occurs, which will lead to a safety related feature or the system being unable to provide the necessary functions for a particular mission objective, such as loss of weapons capability preventing attack. In such cases, the platform is returned to base to undergo repairs. This means that the overall set of missions will cease and an MRP will begin.

For the modelling of abandonment, once the “Phase 1” place is marked, it also marks a “Mission Active” place. If a particular component or sub-system fails, the relevant place representing its failure will be marked, causing an immediate transition to become enabled. This will place a token in the “Mission Abandoned” place. Each phase place has a corresponding immediate transition which, if the mission is active and that particular phase is operational, becomes enabled and places the phase token in the “MFOP Finished” place, to begin the MRP.

## 5. Conclusions

Petri nets provide an effective, easily understood and powerful way of predicting the reliability of a system or platform. The Petri net technique extends to the area of Phased Missions, where complexities of modelling such as component failure rate dependencies, varying distributions and repairable systems are included. The technique can also be used to model a basic Maintenance Free Operating Period, without extensive modelling of many of the expected future technologies which will allow high values of this metric.

The model outlined in this paper can account for various reliability considerations such as component, system, phase, mission and MFOP failure, mission abandonment, the MRP and component failures affecting the failure rate of another component.

## 6. Acknowledgement

The research reported in this paper has been carried out with financial support from BAE Systems and EPSRC.

## 7. References

[1] Esary, J.D., and Ziehms, H., "Reliability Analysis of Phased Missions," *Reliability and Fault-Tree Analysis,* 1975, pp. 213-236.

[2] Burdick, G.R., Fussell, J.B., and Rasmuson, D.M., "Phased mission analysis: a review of new developments and an application," *IEEE Transactions on Reliability,* Vol. R-26, 1977, pp. 43-49.

[3] La Band, R.A., and Andrews, J.D., "Phased mission modelling using fault tree analysis," *Proc. of the 15th Advances in Reliability Technology Symposium (ARTS),* Mech. Eng. Publications for IMechE, 2003, pp. 81-97.

[4] Clarotti, C.A., Contini, S., and Somma, R., "Repairable Multiphase Systems - Markov and Fault-Tree Approaches for Reliability Evaluation," edited by G. Apostolakis S. Garribba and G. Volta, Plenum Press, New York, 1980, pp. 45-58.

[5] Smotherman, M., and Zemoudeh, K., "A non-homogeneous Markov model for phased-mission reliability analysis," *IEEE Transactions on Reliability,* Vol. 33, No. 5, 1989, pp. 585-590.

[6] Smotherman, M.K., and Geist, R.M., "Phased mission effectiveness using a nonhomogeneous Markov reward model," *Reliability Engineering and System Safety,* Vol. 27, 1990, pp. 241-255.

[7] Altschul, R.E., and Nagel, P.M., "The Efficient Simulation of Phased Fault Trees," *Proceedings of the Annual Reliability and Maintainability Symposium,* 1987, pp. 292-296.

[8] Appleton, D.P., "Future Offensive Aircraft - maintenance free operating periods," *Proceedings of the R, M & T for Future Projects Seminar,* 1996.

[9] Hockley, C.J., "Design for success," *Proceedings - Institution of Mechanical Engineers,* Vol. 212, No. G, 1998, pp. 371-378.

[10] Relf, M.N., "Maintenance-Free Operating Periods - The Designer's Challenge," *Quality and Reliability Engineering International,* Vol. 15, 1999, pp. 111-116.

[11] Jones, J.A., Warrington, L., and Davis, N., "Integrated modelling of system functional, maintenance and environmental factors," *Beyond 2001 - The reliability and maintainability odyssey continues; 48th Annual International Symposium on Product and Integrity - The Reliability and Maintainability Symposium (RAMS);* IEEE, Piscataway, NJ, 2002, pp. 399-403.

[12] Warrington, L., Jones, J.A., and Davis, N., "Modelling of maintenance, within discrete event simulation," *Beyond 2001 - The reliability and maintainability odyssey continues; 48th Annual International Symposium on Product and Integrity - The*

*Reliability and Maintainability Symposium (RAMS);* IEEE, Piscataway, NJ, 2002, pp. 260-265.

[13] Petri, C.A., "Kommunikation mit automaten," *PhD Thesis,* 1962,

[14] Malhotra, M., and Trivedi, K.S., "Dependability modeling using Petri-nets," *IEEE Transactions on Reliability,* Vol. 44, No. 3, 1995, pp. 428-440.

[15] Ramchandani, C., "Analysis of asynchronous concurrent systems by timed Petri nets," *PhD Thesis,* 1974.

[16] Beyaert, B., Florin, G., and Lonc, P., "Evaluation of computer systems dependability using stochastic Petri nets," *Digest 11th Annual Symposium on Fault-Tolerant Computing,* IEEE Computer Society, 1981, pp. 79-81.

[17] Molloy, M., "Performance analysis using stochastic Petri nets," *IEEE Transactions on Computers,* Vol. 31, No. 9, 1982, pp. 913-917.

[18] Ajmone-Marsan, M., Conte, G., and Balbo, G., "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems," *ACM Transactions on Computer Systems,* Vol. 2, No. 2, 1984, pp. 93-122.

[19] Volovoi, V., "Modeling of system reliability Petri nets with aging tokens," *Reliability Engineering and System Safety,* Vol. 84, No. 2, 2004, pp. 149-161.

[20] Mura, I., and Bondavalli, A., "Markov Regenerative Stochastic Petri Nets to Model and Evaluate Phased Mission Systems Dependability," *IEEE Transactions on Computers.,* Vol. 50, No. 12, 2001, pp. 1337-1351.