Automatic construction of a reliability model for a phased mission system

This item was submitted to Loughborough University’s Institutional Repository by the/an author.


Additional Information:

- This is a conference paper.

Metadata Record: https://dspace.lboro.ac.uk/2134/17309

Version: Accepted for publication

Publisher: © Loughborough University

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: https://creativecommons.org/licenses/by-nc-nd/4.0/

Please cite the published version.
Abstract

There are a number of mathematical modelling techniques available to determine the reliability of any system design, for example Fault Trees, Event Trees etc. These models relate the performance of the system to the performance of the components of which it is comprised and are generally quite difficult to construct. Once constructed the analysis of the models can be performed using commercially available software. This stage of the analysis is well developed and can be performed efficiently. The model construction however is a lengthy process and reduces the impact of the reliability study on the system design. One way of improving this situation would be to automate the construction process. In this work a procedure is developed to automatically generate a reliability model, based upon Petri Nets, for a system undertaking a phased mission.

1. Introduction

The design stage of any new system is a critical time to ensure that the system meets all required standards. The reliability of the system must be determined to ensure the standards are met. If this can be done alongside the system design in an efficient manner alternative design solutions can be investigated and the direction of the design could be influenced. A number of mathematical modelling techniques exist to determine the reliability of a system, such as fault trees, event trees and Markov analysis etc. These models require not only a detailed understanding of the system design but also understanding of the techniques themselves and hence a specialist group or team is often brought into model the reliability of system designs. This leads to a lack of project cohesion and as the development of the models generally takes a significant amount of time the results obtained are often too late to effectively influence the system design.

Over the years much work has been performed on the analysis of the models once constructed and this is now well developed and can be performed quickly. The area that still involves significant time and effort is the construction of the models. One way of improving this situation is to automate the process, thus enabling the reliability assessment to be performed alongside the design ensuring full use is made of the results. In the past the automatic construction of fault trees has received the most attention in this area. The most commonly adopted approaches include digraphs (Lapp & Powers 1977), decision tables (Salem et al 1977), transition tables (Taylor 1982) and mini fault trees (Kelly & Lees 1986). All these approaches have some form of restriction on their application and so no one method can be applied to all systems. Despite the restrictions on the use of fault trees to
model systems reliability the automation of the other modelling techniques has received little attention in the past. The aim of the work presented here is to outline an approach to automate the generation of a reliability model for a system undertaking a phased mission. Non-repairable and repairable systems have been considered.

A phased mission is defined as a sequence of tasks (phases) which must be completed to achieve the mission objective. For each task to be completed a different sub-set of the system's capabilities need to function. Clearly, the causes of failure will also be different in each phase. For the mission to be completed successfully all of the phases must have been completed successfully. The main techniques that have been used in solving phased mission problems are Fault Tree Analysis, Markov Analysis and simulation. Both Fault Trees and Markov suffer from the issue that the models become very large for such problems and this increases with the complexity of the problem and the number of phases in the mission. Simulation techniques however are well suited to modelling such situations as their computational nature allows for complex scenarios to be considered. One such technique that allows for simple graphical representation as well as significant modelling power is the Petri net.

2. Petri Nets

Petri nets are a graph based tool that can be used to model the dynamics of many types of system, see Schneeweis (1999). Specifically, a Petri net is a directed bipartite graph in which each node represents either a transition or place, shown in diagrams as a bar or hollow circle respectively. Directed arcs linking places to transitions are known as inputs and those connecting transitions to places are known as outputs. In addition, multiple input or output arcs can link the same place and same transition, with the number of arcs known as the multiplicity, often represented as a single arc with a backslash through it and a positive integer denoting the multiplicity. If there is no backslash then the multiplicity is one. Places may contain 0 or more tokens, represented by filled circles, and it is the distribution of tokens through the net, known as the net marking, that determines the state of the system. Each transition is associated and labelled with a time delay which may be fixed or determined from a distribution. When the number of tokens in a place matches or exceeds the number of input arcs, the transition is enabled and will fire once it has remained enabled for the duration of its associated delay, in which case the tokens are consumed from the input places, and deposited in the output places - thus altering the marking of the net and therefore the state of the system. The number of tokens consumed from the input place is equal to the number of input arcs and the number of token deposited in the output place is equal to the number of output arcs. If the marking of the net changes and disables a previously enabled transition, then that transition and its delay duration are reset. Only one transition can occur at any instant of time, regardless of the number of transitions that are enabled. An example of a transition, showing the before and after net markings, is shown in Figure 1.
The figure shows a transition with 3 input places with a multiplicity of 2, 1 and 3 from top to bottom. In the net on the left of the figure the transition is enabled as each input place contains at least as many tokens as its input arcs. Hence after the time delay associated with the transition, t, it fires. Tokens, equal to the number of input arcs, are taken from each input place and tokens equal to the number of output arcs, 2, are deposited into the output place. This is shown in the net on the right of the figure.

3. Automated Model Generation

The aim of the procedure outlined in this work is to take a description of a system and its mission in different phases and to generate a petri net model that will determine the reliability of that system.

3.1 System and mission description

The first stage of the procedure is to input into the system all information that is required to construct and run the model. These can be broken down into the following categories:

1. Component description in the form of component models including the component failure modes,
2. System structure in the form of a system topology diagram,
3. Mission Description in the form of phase models and, initial and starting conditions,
4. System failure modes.
5. Failure and repair data.
6. Maintenance strategies

The component models are in the form of decision tables which describe how the component reacts to inputs from other components in the system, depending on the current state of the component. For example, a bulb which has the 2 states of working (W) and failed (F) would have the decision table shown in Table 1. In the table the inputs (In) and outputs (Out) to the bulb would be connected to other components in the system. In the table, C and NC denote current and no current and the '-' entries are don't care states where the state or input is irrelevant to the output.
Table 1. Decision Table for a bulb in a circuit

Operational mode tables, similar to the state transition tables used by (Majdara & Wakabayashi 2009), are also adopted to model components with different operating modes. These tables describe how the mode of operation can be changed, when a command to the component is introduced. For example, if a switch, which is currently open is commanded by an operator to close, as long as the switch is in a working condition, the switch would change mode from open to closed. The operational mode table for a switch is shown below, where mode 1 is the current mode, In is the input from the operator and mode 2 is the resulting switch mode:

<table>
<thead>
<tr>
<th>Mode 1</th>
<th>In 1</th>
<th>State</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed</td>
<td>-</td>
<td>FCL</td>
<td>Closed</td>
</tr>
<tr>
<td>Closed</td>
<td>CL</td>
<td>-</td>
<td>Closed</td>
</tr>
<tr>
<td>Closed</td>
<td>OP</td>
<td>W</td>
<td>Open</td>
</tr>
<tr>
<td>Closed</td>
<td>NA</td>
<td>-</td>
<td>Closed</td>
</tr>
<tr>
<td>Open</td>
<td>-</td>
<td>FOP</td>
<td>Open</td>
</tr>
<tr>
<td>Open</td>
<td>CL</td>
<td>W</td>
<td>Closed</td>
</tr>
<tr>
<td>Open</td>
<td>OP</td>
<td>-</td>
<td>Open</td>
</tr>
<tr>
<td>Open</td>
<td>NA</td>
<td>-</td>
<td>Open</td>
</tr>
</tbody>
</table>

Table 2 Operational model table for a switch

In the table CL, OP, NA, FCL and FOP denote closed, open, no action, failed closed and failed open respectively.

The system topology diagram describes how the components are linked together. The phase models describe, in the form of a phase transition table, the different phases the mission can enter with the system condition needed to transition from one phase to another. The initial conditions are the conditions the components must satisfy in order for the mission to commence. The failure and repair data is necessary for each component in the system to determine a reliability estimate. The maintenance strategies are the different strategies applied to the components within the system, for example preventative maintenance.

3.2 Model Construction

From the information given, as described in section 3.1, software has been developed that will generate a Petri Net model. The model is made up of different Petri Net types that connect to each other, these types are: Component PN (CPN), Component Model PN (CMPN), Circuit PN (CIPN), System PN (SPN) and Phase PN (PPN).

The component PN’s are simple nets generated from the component description. They link the working and failed states for components and
incorporate the appropriate maintenance strategy. An example of a component PN for a component whose failure is revealed is shown below. The net includes places to record when maintenance is, or is not, taking place hence allowing for maintenance resources to be built into the net and also downtime recorded.

Figure 2. Component Petri net for a component with a revealed failure

For some components the failed state will be dependent upon the current operation mode, an example of such a CPN is shown in Figure 3 for a switch.

Figure 3. Component Petri net for a switch

The Component Model PN’s use the decision tables, operational mode tables and the system topology diagram to generate nets that connect the inputs, states and outputs for the components. For example, considering the decision table for a bulb shown in Table 1 the CMPN is shown below. The places representing the component working and failing, W and F, are connected to the CPN, hence the arrows in the figure.
Figure 4. Component model Petri Net for a bulb

The Circuit PNs represent the flow of current in any electrical circuits that are present and are used to identify, given the state or mode of the components in the system, whether there is current, or not, in the circuit.

The system PN links the CMPNs into the system structure and the Phase PN describes the mission undertaken by the system, created using the phase transition table.

These different nets interact with each other: The CMPN’s link together to form the SPN. The CPN’s pass information to the SPN and the CIPN regarding the states of the components. The SPN pass the information about the current operating modes of the appropriate component to the CPN’s and the CIPN’s. The PPN moves through the different phases of the mission and obtains the information regarding the phase transition requirements from the SPN. If the system makes a transition to a failed state this will be represented by a phase within the PPN and hence failure and reliability information will be output from the PPN.

3.2 Model simulation

The PN’s described above can be used to simulate a systems reliability. Software has been developed to perform this task. The steps undertaken are:

1. From the initial conditions of the system tokens are placed in the relevant places within the CPN’s.
2. Place a token in the place representing the first phase in the PPN.
3. Randomly sample failure and repair times for the components from the relevant distribution.
4. Search through each of the immediate transitions in the CPN, SPN and PPN and determine if any are enabled, if so fire them.
5. If the operating mode of a component changes then check the CIPN’s to determine if the circuits within the system, if there are any, are passing, or not passing, current. This may affect places within the SPN.

When any transition is fired test if any of the following condition are satisfied:

a) a phase transition condition is satisfied. If mission has finished, failure or success, log results and start new simulation.
b) In phase conditions are satisfied for current phase. Check for next timed transition and fire.

3.3 Software

The general structure of the software is shown in figure 5

![Software structure diagram]

Figure 5. Software structure diagram

The information regarding the components: failure information, repair information, maintenance strategy, etc., is entered by the system designers through a text file, alongside the system topology to be modelled. Using this information the software generates the decision and operational mode tables for all the components in the system. The component information is used to generate the CPN’s and the tables are used to generate the CMPN’s and hence the SPN. The component failure and repair information is used to generate the times to failure, $t_F$, and times to repair, $t_R$, that are the time delays for the transitions in the CPN’s. Any circuits in the system are identified by the software by locating all components that are able to pass current and that exist within a path, containing a power supply, that start and end with the same component. Once all the circuits are identified the software automatically generates the CIPN’s.

The description of the mission considered is also input by the designers and the software generates a phase transition table that details the different phases and the system conditions for transition between them. The software then uses this table to generate the PPN.

4. Example

The procedure and software have been applied to a non-repairable pressure tank system, see Stockwell and Dunnett (2013) for more detail. For this non-repairable system the procedure was found to be efficient and to give reliable results. In this current work the procedure has been extended to consider repairable systems. In order to demonstrate the procedure a simple bulb system will be described in detail.

The aim of the system is to light the bulb for a fixed period of time. Initially the system is as shown in Figure 7 with the manual switch open. Power supply 2
(PS2) is a cold standby for power supply 1 (PS1). For ease of presentation here it is assumed that the switching mechanism between the supplies is perfectly reliable. The operator closes the switch and the bulb comes on, after a time $T_1$ hours the operator opens the switch and the bulb goes off. It is assumed that all failures are revealed and if engineers are available maintenance will start immediately.

![Figure 6 Simple bulb system and associated topology diagram](image)

The topology diagram for the system and the decision and operating mode tables for the components are shown in Figure 6 and tables 3-7.

<table>
<thead>
<tr>
<th>Time</th>
<th>In1</th>
<th>State</th>
<th>Out1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t=0$</td>
<td>OFF</td>
<td>W</td>
<td>CL</td>
</tr>
<tr>
<td>$0&lt;t\leq T_1$</td>
<td>OFF</td>
<td>W</td>
<td>NA</td>
</tr>
<tr>
<td>$0&lt;t&lt;T_1$</td>
<td>ON</td>
<td>W</td>
<td>NA</td>
</tr>
<tr>
<td>$t=T_1$</td>
<td>ON</td>
<td>W</td>
<td>OP</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>F</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 3. Decision table for operator

<table>
<thead>
<tr>
<th>In2</th>
<th>State</th>
<th>Out1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>CL</td>
<td>C</td>
</tr>
<tr>
<td>-</td>
<td>OP</td>
<td>NC</td>
</tr>
<tr>
<td>NC</td>
<td>-</td>
<td>NC</td>
</tr>
</tbody>
</table>

Table 5. Decision table for switch.

<table>
<thead>
<tr>
<th>In1</th>
<th>State</th>
<th>Out1</th>
<th>Out2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>W</td>
<td>C</td>
<td>ON</td>
</tr>
<tr>
<td>NC</td>
<td>-</td>
<td>NC</td>
<td>OFF</td>
</tr>
<tr>
<td>-</td>
<td>F</td>
<td>NC</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Table 4. Decision table for bulb.

<table>
<thead>
<tr>
<th>In1</th>
<th>State</th>
<th>Out1</th>
<th>Out2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>W</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>F</td>
<td>NC</td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>-</td>
<td>NC</td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Decision table for power supply

<table>
<thead>
<tr>
<th>In1</th>
<th>In2</th>
<th>Out1</th>
<th>Out2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>NC</td>
<td>NC</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>NC</td>
<td>NC</td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Decision table for junction 1 and 2.
In the case of the operator, the output will depend upon time, this dependence is contained in the phase descriptions and is accounted for in the decision table by adding an extra column for time. The manual switch is the only component considered that has more than one operational mode and hence it also has an operating mode table as given in table 2.

The failure modes of the system have been broken down into ‘failure to start up’, ‘failure to keep bulb alight for $T_1$ hours’, ‘failure to turn bulb off’. In the automation procedure the system failure states have been modelled as separate states and hence there are 6 phases to consider:

- Phase 1: Discrete Phase: System start-up
- Phase 2: Bulb on for duration of $T_1$ hours
- Phase 3: Fail to start up
- Phase 4: System fails to keep bulb lit
- Phase 5: System fails to turn off bulb
- Phase 6: Mission Success

Phases 3-5 are failure states.
From the description the phase transition table can be determined.

<table>
<thead>
<tr>
<th>Time</th>
<th>From Phase</th>
<th>To Phase</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>$B_{out2}=ON$</td>
</tr>
<tr>
<td>$\delta$</td>
<td>1</td>
<td>3</td>
<td>$B_{out2}=OFF$</td>
</tr>
<tr>
<td>$T_1$</td>
<td>2</td>
<td>6</td>
<td>$S_1\ mode=open$</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>4</td>
<td>$B_{out2}=OFF$</td>
</tr>
<tr>
<td>$T_1$</td>
<td>2</td>
<td>5</td>
<td>$S_1\ mode=closed$</td>
</tr>
</tbody>
</table>

Table 8 Phase transition table.

4.1 Petri Nets

As all components are revealed failures the CPN’s will be of the form shown in figure 2. The loop containing the places ‘Maintenance taking place’ and ‘No maintenance taking place’ will be one loop that feeds into all components and will contain the information of whether engineers are available to undertake the maintenance.

The nets for PS1 and PS2 are adapted to account for the fact that the components are in cold standby, see Figure 7. The part of the net inside the large dashed lines corresponds to PS1 and the part inside the dots, PS2. The dashed arrows from the component states feed into the CMPN’s.

The information on the extent and level of redundancy in a system would be included in the detail input by the system designers. A library of CPN’s is being built up for all situations.
Figure 7. CPN’s for PS1 and PS2.

The CMPN’s were generated from the tables 3-8 above and linked together using the system topology diagram to form the SPN which is shown in figure 8. The separate CMPN’s can be identified in the figure by the labels, OP, S, B, J2, PS1, PS2, J1 corresponding to Operator, switch, bulb, junction 2, power supply 1, power supply 2 and junction 1 respectively. The dotted arrows in the figure that input into the components states are passing the information from the CPN’s.
Figure 8. SPN for the bulb system
In this example 2 circuits are identified, one containing PS1 and one containing PS2. The CIPN’s for the 2 circuits are the same with only the place for the power supply changing. The CIPN for circuit 1 is shown in figure 9. The place for ‘current in circuit 1’ will output current to all appropriate inputs to the components in the circuit in the SPN.

Figure 9. CIPN for one of the circuits in the system

The PPN generated from the phase transition table is shown in figure 10. The dotted arrows in this figure connect the places to places in the SPN.

Figure 10. PPN for the mission

Having generated the PN’s the steps outlined in section 3.2 are then undertaken to determine the reliability of the system design.

5. Conclusions
In the work presented here a procedure for automatically generating a reliability model based on Petri nets and simulation, from a description of a system and its operation has been described. The procedure has been demonstrated by applying it to a simple repairable example. The procedure is currently being applied to a more complex repairable system in order to validate its general use.

In the future in order to improve this method of automation, it is hoped that a technique taking the system description in the form of a computer aided design (CAD) diagram or a piping and instrumentation diagram (P&ID) and then generating the component tables will be developed.

References.


