A DYNAMIC FAULT TREE MODEL OF A PROPULSION SYSTEM

Hong Xu/Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia

Joanne Bechta Dugan/ Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia

Leila Meshkat/Jet Propulsion Lab 4800 Oak Grove Drive, Pasadena, CA 91101

SUMMARY/ABSTRACT

We present a dynamic fault tree model of the benchmark propulsion system, and solve it using Galileo. Dynamic fault trees (DFT) extend traditional static fault trees with special gates to model spares and other sequence dependencies. Galileo solves DFT models using a judicious combination of automatically generated Markov and Binary Decision Diagram models. Galileo easily handles the complexities exhibited by the benchmark problem. In particular, Galileo is designed to model phased mission systems.

A phased mission system (PMS) is defined as a system whose mission is composed of multiple, consecutive and non-overlapping phases. Generally, the system configuration, failure criteria and component behavior of different phases may be different. Because the system can change configuration and behavior from phase to phase, dependencies arise between the variables representing components in different phases. The Galileo approach to phased-mission system handles phased mission dependencies as well as dependencies arising from common cause failures, complex redundancy management, shared and cold spares, and other functional dependencies.

Galileo is a software tool for dynamic fault tree developed by the University of Virginia under contract to NASA. Galileo is available commercially from Exelix (exelix.com).

Copyright © #### by ASME

INTRODUCTION

A phased mission system (PMS) is defined as a system whose mission is composed of multiple, consecutive and non-overlapping phases. The reliability of a PMS is defined as the probability that the mission successfully achieves its objectives in all phases. Generally, the system configuration, failure criteria and component behavior of different phases may be different [1]. According to its different characteristics, a PMS can be classified to be static or dynamic, simple or complex with respect to each single phase, identical or different considering structures in different phases, short or long with fewer or more phases.

A static PMS where all phases are static is usually solved using the phase algebra [2, 3] based techniques. While a dynamic PMS where at least one phase is dynamic is solved applying Markov chain (MC) based techniques [4, 5]. There is no problem to handle a simple/small PMS that has two to three phases with each involving just a few components, through these techniques. However, with the increase of the complexity of a single phase structure or the phase number, or both, both PMS-BDD and MC based techniques could face the state space explosion problem. Particularly, when resolving a dynamic PMS, we currently model it by converting all phases into a single MC whose state space is equal to the union of the state spaces of MC associated with each phase [6]. Ou and Dugan have recently developed techniques for modular solution of phased mission systems [13], but this technique has not been fully implemented as yet in Galileo. Without modularization, the single MC constructed for a big PMS could become too huge to handle.

In this paper, we present a dynamic fault tree model of the benchmark propulsion system, which involves 5 phases and each phase consists of 20 modules. One important feature of this system is that the structures or dynamic fault trees (DFT) of all its phases are identical. The only difference between phases comes from the parameters of each component and mission time of a single phase.

BENCHMARK EXAMPLE DESCRIPTION

The original example is an ion propulsion system, which is needed for a science mission to the outer solar system. Totally 17 phases are involved. Table 2-1 lists the mission phases, along with the propulsion system operating time during each phase. For those phases where the propulsion system only operates during part of the phase (e.g., Phases 4, 10, and 14), thrust is continually provided from the beginning of the phase until the specified operating time expires.

Table 2-1. Propulsion System Mission Profile

Mission Phase	Duration (hours)		Propulsion System Operating Time (hours)	
1	5520.0		5520.0	
2	336.0		0	
3	9043.2		9043.2	
4	26280.0		13140.0	
5	**13140.0**		**13140.0**	
6	**5040.0**		**5040.0**	
7	**1483.2**		**1483.2**	
8	**720.0**		**720.0**	
9	**1444.8**		**1444.8**	
10	4315.2		2880.0	
11	1588.8		1588.8	
12	720.0		0	
13	1540.8		1540.8	
14	4327.2		2880.0	
15	1101.6		1101.6	
16	720.0		0	
17	720.0		720.0	

The propulsion system consists of ten thruster assemblies and a propellant supply. Each assembly has one propulsion power unit (PPU) and two ion engines. When an assembly is operating, the PPU provides power to just

Copyright © #### by ASME

one ion engine. The other engine will be in a standby mode, unless failed. Figure 2-1 is a schematic of a thruster assembly. Ion Engine A will continue to be the operating engine of the assembly until the engine fails.

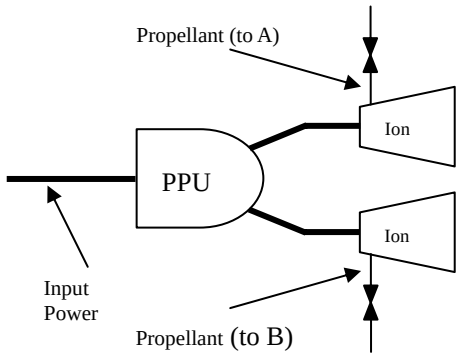


Figure 2-1. Thruster Assembly Schematic

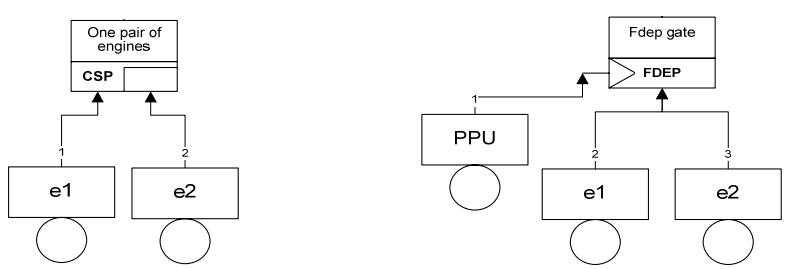
During Phase 1 the success criterion is propulsion from four assemblies. During subsequent phases, where the propulsion system is operating, the success criterion is propulsion from eight assemblies. In assessing the mission risk input power failures are modeled separately, so the propulsion system model can ignore a loss of power from that support system. The strategy for thruster assembly operation is to begin with power from the PPU going to Ion Engine A. Ion Engine A will continue to be the operating engine of the assembly until the engine fails. At that time the strategy is to (1) shutdown the PPU, (2) switch the PPU to Ion Engine B, (3) reenergize the PPU and operate with Ion Engine B. There are no intermediate switches between a PPU and the ion engines. All switches are integral to the PPU.

DERIVED MODEL

In order to state our method clearly (and to facilitate manual verification of results), we derive a modified example from the original one. The derived example has 5 phases, which are the same as the original phases 5 through 9, marked to be bold in Table 2-1 above. Furthermore, we extend the number of the thruster assemblies from 10 to 20 so that it is more convincing to demonstrate the feasibility of our method with respect to complex systems.

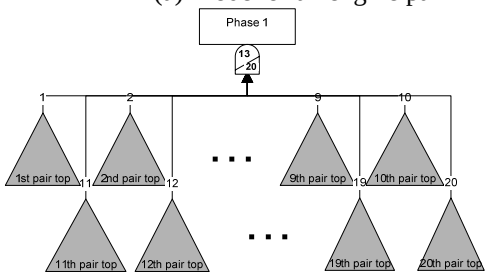
Considering that each assembly contains 2 engines, with one operational and one standby, it is reasonable to model such a structure via a dynamic cold spare gate (CSP) [7, 8], which is shown in Figure 3-1(a). We use a dynamic functional dependent gate (FDEP) to model the behavior of PPU unit, the failure of which will directly lead to the failures of both engines. Figure 3-1(b) depicts the model of the dependency between a pair of engines and their corresponding PPU unit.

From Figure 3-1(c), we know that phase 1 is composed of 20 inputs, each of which is represented by a transfer gate. One transfer gate here is actually equal to an assembly module. The top gate in Figure 3-1(c), a static K-out-of-M gate with K = 13 and M = 20 tells us that the failures of any 13 or more assemblies will bring down the system. This is equivalent to the success criteria which requires that at least 8 assemblies be operational. Based on Figure 3-1(d), we can see that a failure of any single phase will result in the top event, the failure of whole system. A transfer gate in Figure 3-1(c) representing a single assembly is therefore composed of a PPU and two engines, which is already shown in Figure 3-1(a) and (b). Since it is supposed in this report that all phases have identical structures, the model in Figure 3-1(c) for phase 1 can be also applied to any other phases other than phase 1. Thus, an integrated model is further constructed in Figure 3-2.

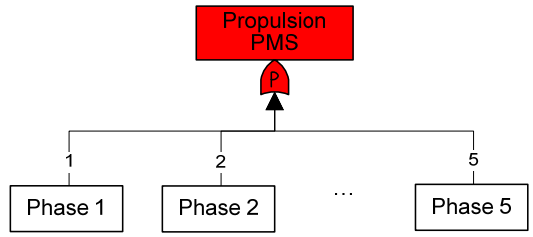


Copyright © #### by ASME

(a) Model of an engine pair



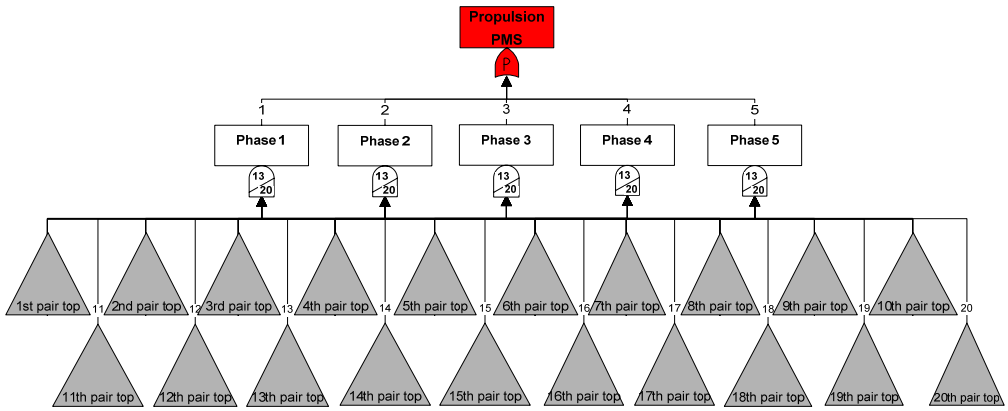
(b) Model of dependency



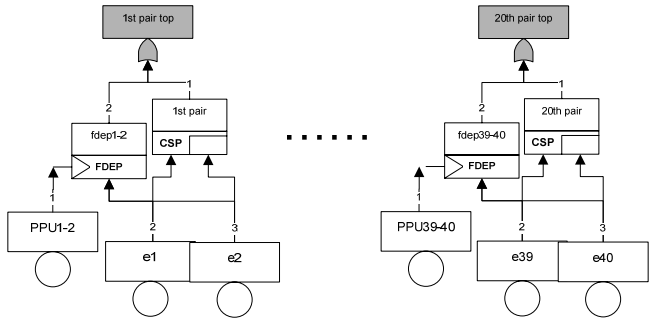
(c) Model of a single phase

(d) Model of the whole propulsion PMS

Figure 3-1. Model of the derived example



(a) Model of the main page



(b) Model of the sub-page w.r.t. each assembly module

Figure 3-2. An integrated model of the propulsion example

BENCHMARK EXAMPLE ANALYZING

During the analysis process, we faced two primary difficulties. One difficulty comes from dealing with dynamic multiple phases, the other difficulty comes from dealing with a static K-out-of-M gate. In this section we will discuss these issues and our corresponding strategies.

Dealing with Dynamic Multi-phases

Normal Way

When analyzing a dynamic PMS, the normal way is to consider the system as a whole and accordingly to construct a single MC which covers the state space of all phases of the system. Let us first look at a simple dynamic PMS called PMS1, shown in Figure 4-1. PMS1 includes 2 phases, each having 2 modules, First pair top and Second

Copyright © #### by ASME

pair top. Either module's failure will lead to the top event's failure. Using the normal way, which means that no modularization will be performed all basic events appearing in any phase are collected to construct the single MC of the whole system. Each time when creating a new destination state from a source state, a thorough phase-by-phase check is needed in order to ensure that the new state be not a duplication of an existing state, and also to ensure that the new state be valid for at least one phase. After all MC states are created, the corresponding transitions are also created for all phases. Figure 4-2 provides the MC of PMS1.

For such a simple system, there are 5 states and 32 transitions involved in total. It seems that the MC in Figure 4-2 is easy to handle. However, with the increase of complexity of each single phase or the number of phases, the corresponding MC will become very huge so that be hard to resolve. For instance, based on the model in Figure 4-1, suppose that only one more phase and one more similar module are added. Then the updated MC will be inserted 19 more states and 112 more transitions. We will give a concrete comparison between this normal way and our alternative in later section.

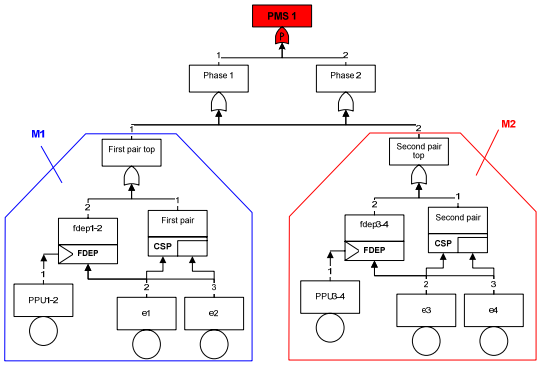


Figure 4-1. Model of dynamic system PMS1

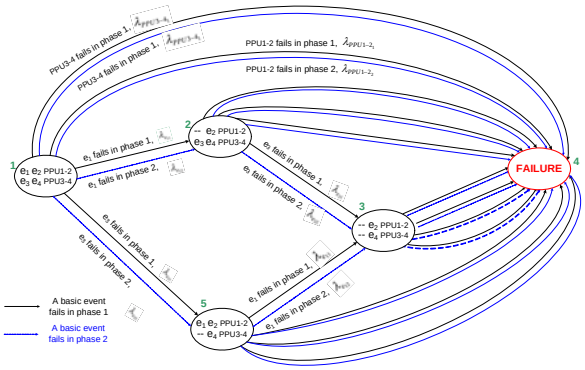


Figure 4-2. MC of dynamic system PMS1

Alternative Way

By contrast to the normal way to treat a PMS, we note that all phases have the same DFT structures then share the same MC state space as phase 1, which allows possible to only analyze the DFT and MC of phase 1. Since modularization within a non-phased system (or a single phase system), is much easier to carry out and also has been implemented in some reliability analysis tools, the problem to handle multiple dynamic phases is then converted to an easy-handle issue related to a single phase. The main three steps are listed here. (1) To identify two modules M1 and M2 within phase 1, shown in Figure 4-1. (2) To construct the MC for each module within phase 1. In addition to the transitions that are related to phase 1, all transitions related to any other phases are created in this MC. Figure 4-3 illustrates the MC of module M1 in phase 1. (3) To resolve each MC by summing up the probabilities of all failed states within one certain phase to get the failure probability of that phase, then summing up the failure probabilities of all phases to get the system unreliability. For more details about calculating a phased-mission MC, refer to [9].

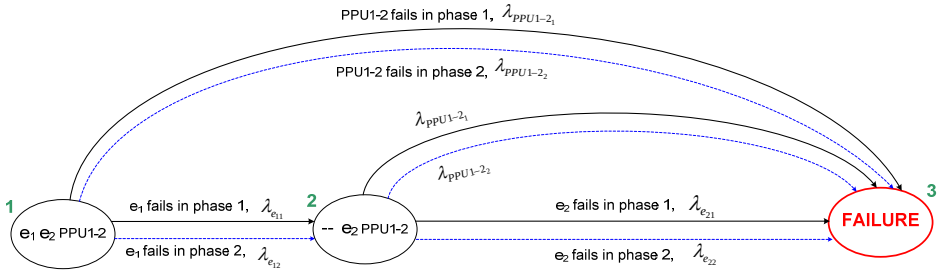


Figure 4-3. MC of module M1 in phase 1

Figure 4-3 indicates that module M1 includes 3 states and 8 transitions. Module M2 also has such a similar MC structure. Comparing to the normal way, the number of transitions with the alternative way decreases by 50% (from 32 to 16), while the number of states actually increases a little bit (from 5 to 6). But the efficiency of this alternative way can be particularly demonstrated in analyzing more complex PMS with structure-identical phases. Consider the same updated example mentioned in section 4.1.1, which contains 3 phases and 3 modules. With the alternative

Copyright © #### by ASME

way, the states and the transitions only increase linearly, from 6 to 9 and from 16 to 36, respectively. In Table 4-1, we make a comparison of these two ways in terms of unreliability and time cost [(1)].

Table 4-1. A comparison between the normal way and the alternative way [(2)]

PMS Structure		Normal Way		Alternative Way	
Modules	**Phases**	**Unreliability**	**Time cost (sec.)**	**Unreliability**	**Time cost (sec.)**
2	2	0.003063605247	**2.141**	0.003063605247	**1.250**
3	2	0.004697148430	**7.954**	0.004697148430	**2.078**
4	2	0.006444326206	**40.100**	0.006444326206	**3.141**
5	2	0.003866119080	**6267.082**	0.003866119080	**5.953**

It is observed from Table 4-1 that with the normal way, the time cost increases dramatically from a few seconds to more than one hour when the number of modules is changed from 2 to 5. By contrast, with our alternative way, the time cost only increases a little bit without losing the correctness of the results.

Figure 4-4 reflects an intrinsic difference between these two ways. From this Figure, we can see that with the normal way, the time cost increases suddenly after the number of modules reaches 3. There is no doubt that it would be very difficult even impossible to resolve a more-phase or more-module system through the normal way. However, with the alternative way, the curve increases much slower. Besides the data in Table 4-1, a third curve is also illustrated in Figure 4-4, which reflects the relationship between the time cost and the number of modules of a 5-phase PMS. When the number of modules is up to 20, the time cost is still less than 100 seconds or 2 minutes.

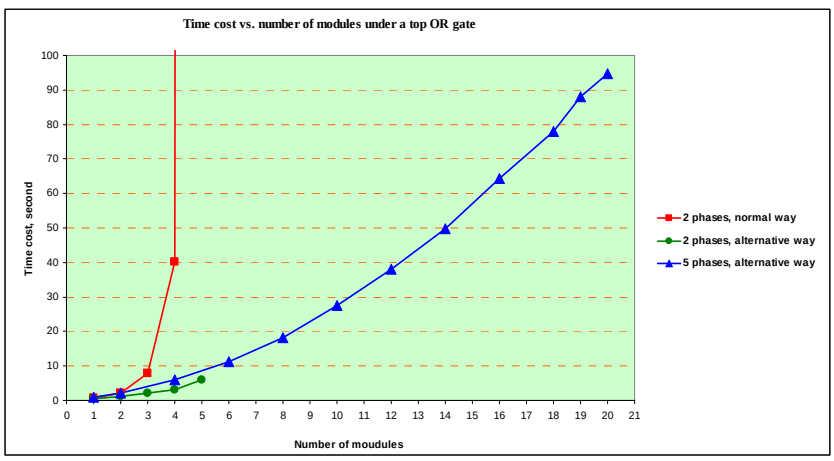


Figure 4-4. Time cost versus the number of modules in two ways

Dealing with Static K/M Gate

After we overcame the difficulty caused by multiple identical dynamic phases, another new difficulty is put in front of us – when we go back to handle the derived example shown in Figure 3-2, a problem of short-of-memory appeared. This problem has nothing to do with dynamic issue but with static K-out-of-M or K/M gates.

A K/M gate has a strong modeling ability, which is often used in a system with redundancy. When dealing with a K/M gate, a regular method is to expand it into a set of AND and OR gates. For example, in Figure 4-5, an original static system with a 2/3 gate is expanded into a new system with one OR gate and three AND gates, each connecting two inputs. The expansion process implies that a K/M model expands the size of static fault tree (SFT) structures then the binary decision diagram (BDD) space in later analysis, which makes harder to handle a big PMS. Generally, a K/M gate will be expanded into C_M^K combinations of inputs. Therefore the 13/20 gate in Figure 3-2 will be expanded into 77520 AND gates and 1 OR gate. This would be obviously a disaster!

Copyright © #### by ASME

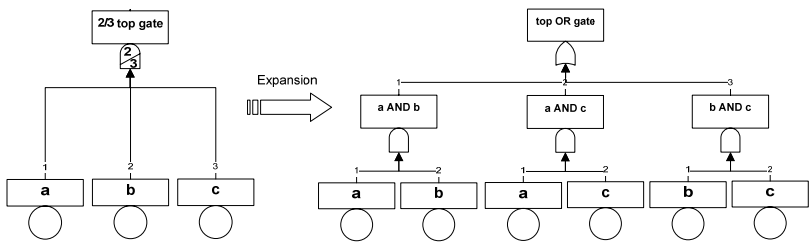


Figure 4-5. Expansion process of a system with 2/3 gate

In this paper, we adopt a recursive algorithm to avoid such an expansion [10 - 12]. The main idea is to make use of the Shannon decomposition theory as shown in Eq. (4-1). Suppose there is a static system with a K/M gate, in which there are *M* inputs and *K* or more input failures will result in the system failure. In this recursive algorithm, once a certain single input fails, we need to consider whether *K-1* or more inputs fail in the remaining *M-1* inputs; if this certain single input is operational, we need to consider whether K or more inputs fail in the remaining *M-1* inputs. Let f_M^K represent the Boolean function of the top K/M gate of a system, x_i and $\overline{x}_i$ representing the facts that the *i-th* input is failed and operational, respectively. Thus the Shannon decomposition expression above can be rewritten as Eq. (4-2) for this K/M system.

$$f = x_i \cdot f_{x_i=1} + \overline{x}_i \cdot f_{x_i=0} \qquad (4-1) \qquad\qquad f_M^K = x_i \cdot f_{M-1}^{K-1} + \overline{x}_i \cdot f_{M-1}^K \qquad (4-2)$$

With such an iterative expression, we successfully avoid the risk to expand a K/M system thus also reduce the memory usage efficiently. Figure 4-6 and 4-7 reflect some satisfying results [(3)]. Figure 4-6 considers a 5-phase system with *M = 20* modules (referred to the example in Figure 3-2) with a K/M top gate and *K* varying from 1 to 13. Figure 4-6 shows that when the number *M* is fixed, the increase of number *K* has little effect on the time cost. In this Figure, the time cost is nearly invariable, which is between 1 minute and 2 minutes. In Figure 4-7, we set each single phase to be a fixed 13/20 structure, and increase the number of phase(s) from 1 to 5. The almost straight line in Figure 4-7 indicates that the ratio of the time cost to the number of phases is close to a constant, which means that under the recursive algorithm and the alternative way, the time cost increases approximately linearly with the increase of the number of phases.

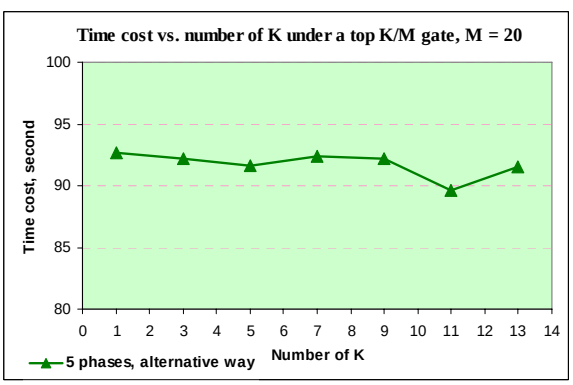


Figure 4-6. Time cost versus the number of K for a K/M system

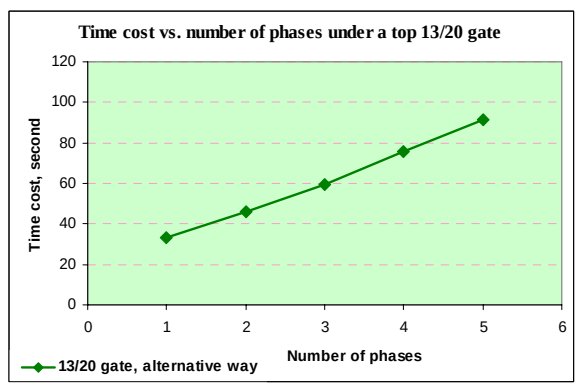


Figure 4-7. Time cost versus the number of phases for a 13/20 system

Final Result

After resolving the two aspects of the primary difficulties, it is now the time to solve the derived example in Figure 3-2. Here we modified the original parameters of *"ei"* a little bit, and select the mission times of phases 5 through 9 from Table 2-1. All data are listed in Table 4-2. The final system unreliability is **2.48e-013**, which costs **89** seconds (about one minute and a half).

We note that this unreliability is very small, and the parameter values we used were adapted from those provided by NASA. In particular, the fail to switch was changed from "per demand" to "per hour." An accurate

Copyright © #### by ASME

dependability analysis must consider the availability of the system on demand, and the reliability of the system during demand. Meshkat & Dugan [14] have developed an approach for the automatic combination of these analyses within the context of dynamic fault trees. (We note that funding cuts prevented the implementation of the methodology needed to consider 'per demand' occurrences.)

Table 4-2. Reliability Data

Component Type	Failure Mode	Failure Rate Value
PPU	Fails to operate (per hour)	1.0×10^{-6} in all phases
Ion Engine / ei	Fails to operate (per hour)	1.0×10^{-5} in phase 1
		1.2×10^{-5} in phase 2
		1.3×10^{-5} in phase 3
		1.4×10^{-5} in phase 4
		1.5×10^{-5} in phase 5
Mission time (hour)	Phase 1	13140.0
	Phase 2	5040.0
	Phase 3	1483.2
	Phase 4	720.0
	Phase 5	1444.8

(1) All results are obtained from our reliability analysis tool, Galileo.

(2) The data in Table 2-1 and Figure 4-3 are based on the following parameters:

Table 4-3. Parameters used for Figure 4-3

Mission Time	phase 1	phase 2	phase 3	phase 4	phase 5
Failure Rate	10	20	30	40	50
e1	1.0E-04	1.2E-04	1.3E-04	1.4E-04	1.5E-04
e2	2.0E-04	2.2E-04	2.3E-04	2.4E-04	2.5E-04
e3	3.0E-04	3.2E-04	3.3E-04	3.4E-04	3.5E-04
e4	4.0E-04	4.2E-04	4.3E-04	4.4E-04	4.5E-04
e5	5.0E-04	5.2E-04	5.3E-04	5.4E-04	5.5E-04
e6	6.0E-04	6.2E-04	6.3E-04	6.4E-04	6.5E-04
e7	7.0E-04	7.2E-04	7.3E-04	7.4E-04	7.5E-04
e8	8.0E-04	8.2E-04	8.3E-04	8.4E-04	8.5E-04
e9	9.0E-04	9.2E-04	9.3E-04	9.4E-04	9.5E-04
e10	10.0E-04	10.2E-04	10.3E-04	10.4E-04	10.5E-04

(3) The data in Figure 4-5 and Figure 4-6 are based on the following parameters:

Table 4-4. Parameters used for Figure 4-5 and Figure 4-6

	phase 1	phase 2	phase 3	phase 4	phase 5
mission time	10	20	30	40	50
ei	1.0E-04	1.2E-04	1.3E-04	1.4E-04	1.5E-04
PPU	0.5E-4	0.5E-4	0.5E-4	0.5E-4	0.5E-4
Number of K	**Unreliability**			**Time cost （second）**	
1	1.428187201E-01			92.670	
3	4.674808054E-04			92.205	
5	3.752153089E-07			91.641	
7	1.114836387E-10			92.421	
9	1.439310982E-14			92.202	

Copyright © #### by ASME

11	8.588159316E-19	89.592
13	2.366967122E-23	91.561
Phase NO.	**Unreliability**	**Time cost （second）**
1	9.524395682E-39	32.999
2	1.555120822E-32	45.781
3	1.332259270E-28	59.516
4	1.095798012E-25	75.640
5	2.366967122E-23	91.561

ACKNOWLEDGMENTS

Galileo is a software package capable of analyzing the reliability of complex phased mission systems using Dynamic Fault Trees. Galileo is available to NASA personnel (contact Duane.H.Pettit@nasa.gov) or commercially from Exelix (info@exelix.com).

The research described in this paper was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

REFERENCES

[1] Xing, L. and Dugan, J. B., 2000, "Reliability analysis of static phased mission systems with imperfect coverage," *Proceedings of The Second International Conference on Mathematical Methods in Reliability*.

[2] Zang, X. and Sun, H. and Trivedi, K. S., 1999, "A BDD-based algorithm for reliability analysis of phased-mission systems," *IEEE Transactions on Reliability*, **48(1)**, pp. 50-60.

[3] Xing, L. and Dugan, J. B., 2002, "Analysis of generalized phased-mission system reliability, performance, and sensitivity," *IEEE Transactions on Reliability*, **51(2)**, pp. 199–211.

[4] Alam, M. and Al-Saggaf, U. M., 1986, "Quantitative reliability evaluation of repairable phased-mission systems using Markov approach," *IEEE Transactions on Reliability*, **R-35(5)**, pp. 498–503.

[5] Mura, I. and Bondavalli, A., 2001, "Markov regenerative stochastic petrinets to model and evaluate phased mission systems dependability," *IEEE Transactions on Computer*, **50(12)**, pp. 1337–1351.

[6] Dugan, J. B., 1991, "Automated Analysis of Phased-Mission," *IEEE Transactions on Reliability*, **40(1)**.

[7] Dugan, J. B., Bavuso, S. and Boyd, M., 1992, "Dynamic fault tree models for fault tolerant computer systems," *IEEE Transactions on Reliability*.

[8] Dugan, J. B., Bavuso, S. and Boyd, M., 1993, "Fault trees and Markov models for reliability analysis of fault tolerant systems," *Reliability Engineering and System Safety*, **39**, pp. 291–307.

[9] Xu, H., 2004, "Combining Dynamic Fault Trees and Event Trees for Probabilistic Risk Assessment," M.S. thesis, University of Virginia, Charlottesville.

[10] Rushdi, A. M., 1986, "Utilization of Symmetric Switching Functions in the Computations k-out-of-n Systems Reliability," *Microelectronics and Reliability*, **26(5)**, pp. 973-987.

[11] Rushdi, A. M., 1991, "Common on: An Efficient Non-recursive Algorithm for Computing the Reliability of k-out-of-n Systems," *IEEE Transactions on Reliability*, **40(1)**, pp. 60-61.

Copyright © #### by ASME

[12] Tang, Z., 2002 “Common Cause Failure Analysis and Improved Solution Techniques for Dynamic Fault Trees”, M.S. thesis, University of Virginia, Charlottesville.

[13] Yong Ou and Joanne Bechta Dugan, “Modular Solution of Dynamic Multi-Phase Systems,” IEEE Transactions on Reliability, December 2004, pages 499-508

[14] Meshkat, L., J.B. Dugan, & J.Andrews "Dependability analysis of systems with on-demand and active failure modes, using dynamic fault trees” IEEE Transactions on Reliability, (20020600) vol. 51,no 2, pp. 240-251; Summer 2002

Copyright © #### by ASME