World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:5, 2008

A Novel Methodology for Synthesis of Fault Trees from MATLAB-Simulink Model

F. Tajarrod, and G. Latif-Shabgahi

***Abstract*—**Fault tree analysis is a well-known method for reliability and safety assessment of engineering systems. In the last 3 decades, a number of methods have been introduced, in the literature, for automatic construction of fault trees. The main difference between these methods is the starting model from which the tree is constructed. This paper presents a new methodology for the construction of static and dynamic fault trees from a system Simulink model. The method is introduced and explained in detail, and its correctness and completeness is experimentally validated by using an example, taken from literature. Advantages of the method are also mentioned.

***Keywords*—**Fault tree, Simulink, Standby Sparing and Redundancy.

I. Introduction

FAULT tree is widely used in the reliability and safety assessment of engineering systems for over 40 years. The fault tree forms a basis for the analysis of a system primary design, and assuring system non-functional requirements (such as reliability, availability and safety). A fault tree is a graphical representation of the relations between basic failures and a specific undesired system failure (Top event). The main result of a fault tree analysis is the probability of occurrence of the Top event. The procedure consists of two steps; construction of the fault tree, and analysis of the tree. In the first step the tree is constructed from a model of real system, automatically, or manually. In the next step, the constructed fault tree is analyzed, qualitatively or quantitatively.

A. Fault Tree Construction Methods

Taking into account the high complexity of today's technical systems, manual construction of fault trees is, in many cases, not possible within the limits of existing time and budget constraints. Manual construction is a time-intensive, costly, laborious, usually incomplete, and prone to errors-of-omission and inaccuracies. Automatic generation of fault tree permits saving time and effort, and increases the quality of results. Completeness, correctness, and consistency of the generated fault tree are also guaranteed [1] Automatic fault tree generation allows the analyst to concentrate on system definition, speeds up the analysis and verification of complex systems [2], and provides faster risk analysis [3]. It ensures the uniform handling of the different system variants, and thus it makes the exhaustive analysis of the system within a short period of time leading to reduce costs associated with manual fault tree analysis. So, in the design phase the system plan can be modified based on the on-line results of the safety analysis.

A variety of construction techniques have been introduced in the literature; each has some strong and some weakness points. Reference [4] developed a fault tree construction methodology for electrical systems from the failure transfer function of system components. This method was later modified by a number of authors. Reference [5] improved the method to deal with control loops in the system by using state-transition table of system. The authors of [6] used digraphs with operators to cope with control loops. Decomposition of a plant into a set of control loops to construct fault trees was then presented in [7]. The fault tree synthesis algorithm of [8] was based on a mini-fault tree. These mini-fault trees are generated from propagation equation, event statements, and decision tables. The 'relations' between process variables and component states to model the fault propagation in the system has been used in [9]. Reference [10] developed a quantitative procedure to build fault tree from a schematic diagram of electrical systems. In [11] another methodology based on extended decision tables of system components was presented. Reference [12] presented a rule-based approach to construct a system fault tree from its reliability block diagrams. In the last decade, some useful methods have been introduced to the construction of fault trees from a system topology diagrams. This type of methods avoid the tedious work of generating decision tables, failure transfer functions, state transition tables, diagraphs, bigraphs and etc. The works explained in [13] and [14] are example of this type of methods. The main advantage of these approaches is that the established models within the library can be reused in different applications and can be easily modified or extended to new component models.

This paper introduces a novel methodology for the construction of a system fault tree. The starting model is the Simulink-model of the real system. The main advantage of Simulink representation of a system topology is its widely application in the modeling and simulation of engineering systems. The construction of Simulink-based fault tree has not been addressed in the literature; however, some useful comments and ideas have been explained in [15]. The construction method will be adequately explained and the procedure will be introduced through examples. Of the advantages of our methodology are: the flexibility, scalability, and its modularity to other popular methods; all of these are

Manuscript received April 28, 2008.

F. Tajarrod and G. Latif-Shabgahi are with the Computer group, Electrical Engineering Department, Power and Water University of Technology, Tehran, Iran (e-mail: f.tajarod@Gmail.com).

explained in section V. The organization of this paper is as follows. In section II the novel construction methodology is introduced. Section 3 gives an example for the proposed method. Section IV lists the advantages of our methodology. Finally, some concluding remarks are given in section V.

ABBREVIATIONS

CSP: Cold Spare gate
DFTA-code: The name of our dynamic fault tree analysis program
FDEP: Functional Dependency gate
FTC-code: The name of our fault tree construction program
HSP: Hot Spare
MAS: Mission Avionics System
MofN: M out of N gate
PAND: Priority-And gate
SFTA-code: The name of our static fault tree analysis program
VM: Vehicle Management

II. THE NOVEL METHOD

A. System Topology

In this method, the system is first defined in the Simulink environment. There is no limitation for a user in the designing of system model. The user can take benefit all of the capabilities of Simulink. Two points should be considered in the implementation of the model:

- Each component and sub-system should have a unique name.
- Components or sub-systems performing N distinct functions (multi-function components), are considered as N virtual separate components. This enables the user to define the whole system as a collection of single-failure components. For example, the failing of a component with two functions A and B, might be the result of the failure of either A or B or both. It is obvious that failure impact of A is different from that of the B.

Multi-function components are identified by counting the number or variety of their inputs or outputs. A component with 2 distinct outputs or 2 distinct inputs is a double-function component.

Once the Simulink model is constructed, the system description (defined below) and the user-defined Top event are taken for building the Extended model of the system.

The Extended model is then used to generate the FAULT TREE diagram of the system. The tree is then analyzed by using the failure probability (P) or failure-rate probability (λ) of basic events.

Fig. 1 illustrates the flow-chart of the method. In section II-A the detailed description of segment A (of the Fig. 1), and the detailed explanation of segment B will be presented in other paper of these authors, elsewhere.

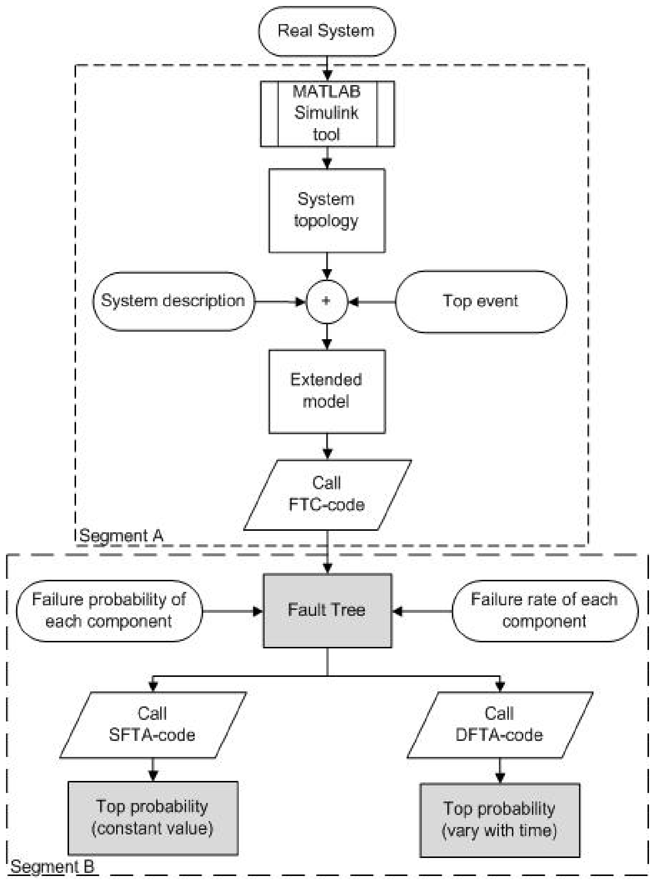


Fig. 1 Flow-chart of the method

B. Fault Tree Construction

Having constructed the system topology in Simulink, the user defines the top event, and enters the functional and behavioral information of the system to build its Extended model. The Extended model determines that i) which components have no impact on the occurrence of Top event, ii) which set of components or subsystems co-operate toward a special aim, and iii) which components or subsystems support each other toward a specific goal. The following steps are used to the construction of the Extended model.

FIRST STEP: IDENTIFICATION OF COMPONENTS

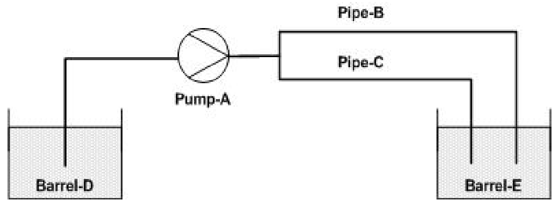
Once the Top event is specified, the type of all system components must be identified. We have classified components into 7 groups; “no-affected”, “affected”, “effected”, “primary”, “alternative”, “priority”, and “usual” components. Components that their failures have not impact on the failing of the whole system or the failure of other components are considered as “no-affected” type components. A component, for which there is no interest for its impact on the Top event, is also considered as “no-affected” component. A component that its failure impacts another component (and makes it failed or unusable) is called “affected” component. For example, the trigger component of a FDEP gate of dynamic fault trees [16] is of this type. Components that their failure is occurred due to the failure of another component are considered as "effected" components. For example, the dependent component of a FDEP gate is of this type. A basic component with cold-sparing gate is labeled as "primary" component. And the other non-primary components of a cold-sparing gate are labeled as "alternative" components. A component with ith "priority" in a subsystem consisting of i components is the one whose failure after the occurrence of (i-1) failures will cause the whole subsystem failures. Other effective components on the Top event are labeled "usual" components.

SECOND STEP: IDENTIFICATION OF SUBSYSTEMS

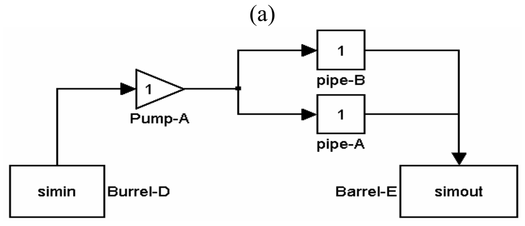
Once the components are labeled, the subsystems should be determined. A subsystem is a group of components to perform a specific task. Two types of subsystems are defined: "redundant" and "usual". A "usual" subsystem consists of co-operating components toward a common function (e.g. Trip Valve). And a "redundant" subsystem is the one consists of replicated or supporting components. "Redundant" subsystems are classified into 3 groups: "M-of-N", "CSP" and "PAND".

After the identification of components and subsystems, the Extended model is constructed. This model is now ready to be delivered to our program to generate the fault tree model of the system in the Simulink window. The generated tree can be resized, moved and reshaped based on the user's wish. It can be also copied and used as a sub-tree in another tree. The tree can also be edited directly from MATLAB command line.

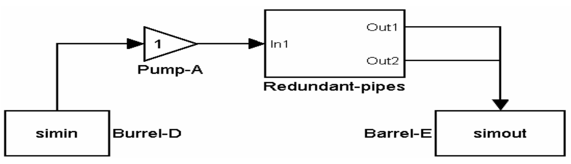
These steps are shown for a simple mechanical system in Fig. 2.a. The Top event, here, is "no-fuel into barrel E". Fig. 2.b indicates the block diagram of this system in the Simulink environment. Fig. 2.c shows the Extended model of this system, constructed by using the above-mentioned steps. The generated fault tree is indicated in Fig. 2.d.



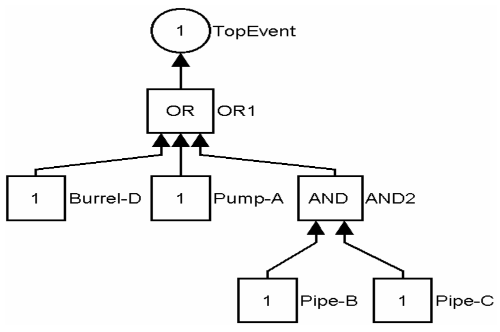
(a)



(b)



(c)



(d)

Fig. 2 a) A mechanical system. b) System Simulink model. c) Extended model d) Constructed fault tree

III. A CASE STUDY

In this section, the "Mission Avionics System" taken from [17] is studied for presenting and illustrating our methodology. Fig. 3 shows the original system model of MAS. The fault tree of the system will be a dynamic fault tree and hence this is a good example for presenting the purposed methodology.

The main components of this system are:

- vehicle management component,
- crew station control & display,
- mission & systems management (mgmt),
- local path generation,
- Scene & obstacle control.

One processing unit is required for the crew station functions, mission and system management, and local path generation. Each of these units is supplied with a HSP back-up to take over control to detect an error. For example, mission & system management has a primary unit, called system mgmt a, and an active HSP back-up, called system mgmt b. The scene and obstacle and VM units both require more functionality than 1 processing units, each of which also has an active back-up.

In addition to the HSP backups, two additional pools of spares are provided, each containing two CSP processing units. Spare1 and Spare2 can be used to cover the failure of the first

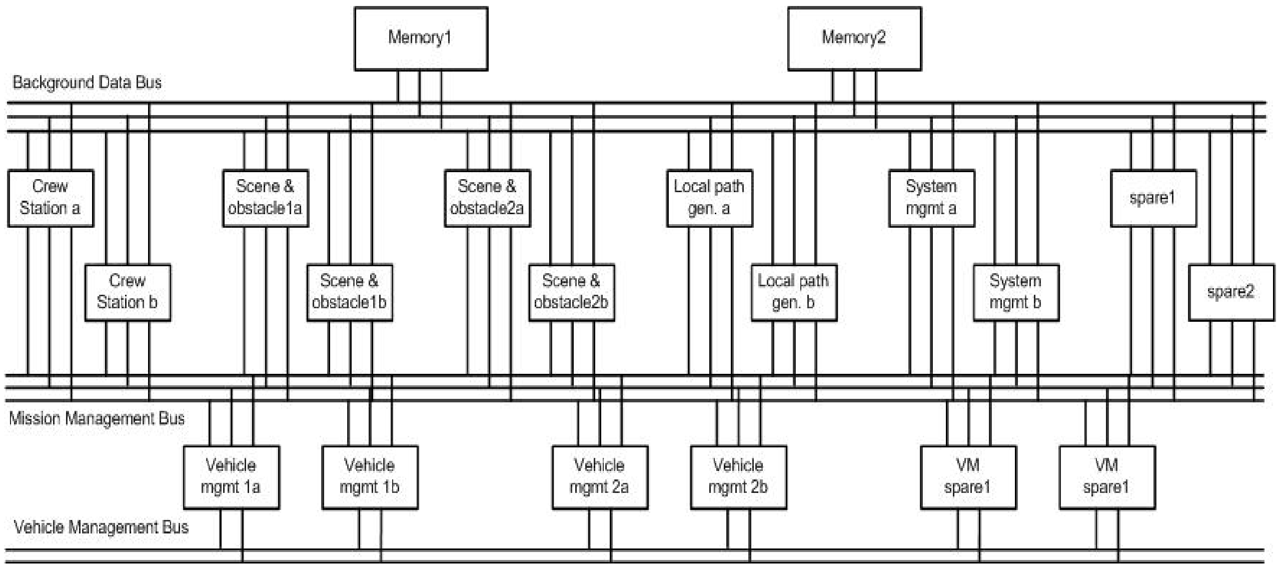


Fig. 3 Mission Avionics System

2 processors in the units other than the VM.VM Spare1 & VM Spare2 cover the failures in the VM unit. The units are connected by 2 triplicated bus systems; the first is the data bus and the second is the mission management bus. The VM has an additional duplicated bus, the vehicle management bus. The system fails in the following three cases:

- any of the systems cannot perform the functions, or
- both the memories fail, or
- all the busses in any one type fail.

MATLAB-Simulink model of MAS system is shown in Fig. 4.

To construct the dynamic fault tree model of this system the above mentioned steps are performed. All Buses and Memories are labeled as usual components. 'vehicle management component', 'crew station control & display', 'mission & systems management', 'local path generation' and 'scene & obstacle control' are the primary components of CSP gate, thus they are labeled as 'Primary' components.

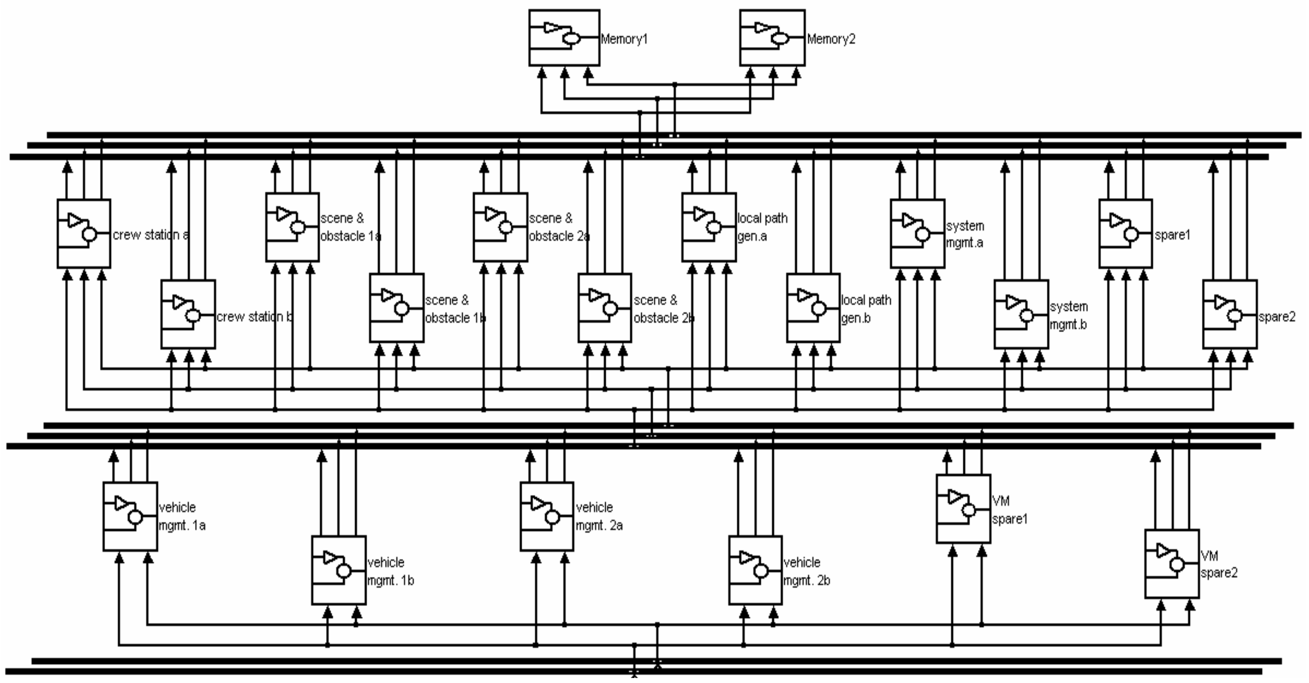


Fig. 4 MATLAB-Simulink model of MAS system

The remaining components are of type 'alternative'; 'VM spare1' and 'spare1' are named alternative1, and 'VM spare2' and 'spare2' are named alternative2 in this modeling. Next, we should built subsystems. 'Memory1' and 'Memory2' are assumed as a redundant subsystem of type MofN. 'Background data buses' build the next subsystem, 'Mission management buses' make another subsystem, and 'vehicle management buses' build the next subsystem. These three subsystems are redundant of type MofN. Any primary component along with the two alternative components built a unique CSP subsystem. These two alternative components are copied and build another CSP subsystem with the next primary component. This is done for all of the existing primary components. Two consecutive CSP subsystems are labeled as a redundant subsystem of type MofN.

Fig. 5 clarifies the construction method of the Extended model. Two CSP subsystems that construct a MofN subsystem are shown in the right hand side of the Fig. 5. Other parts of the system are obvious. Fig. 6 shows the Extended model of the system. The fault tree model of the system which has been constructed from Fig. 6 is shown in Fig. 7.

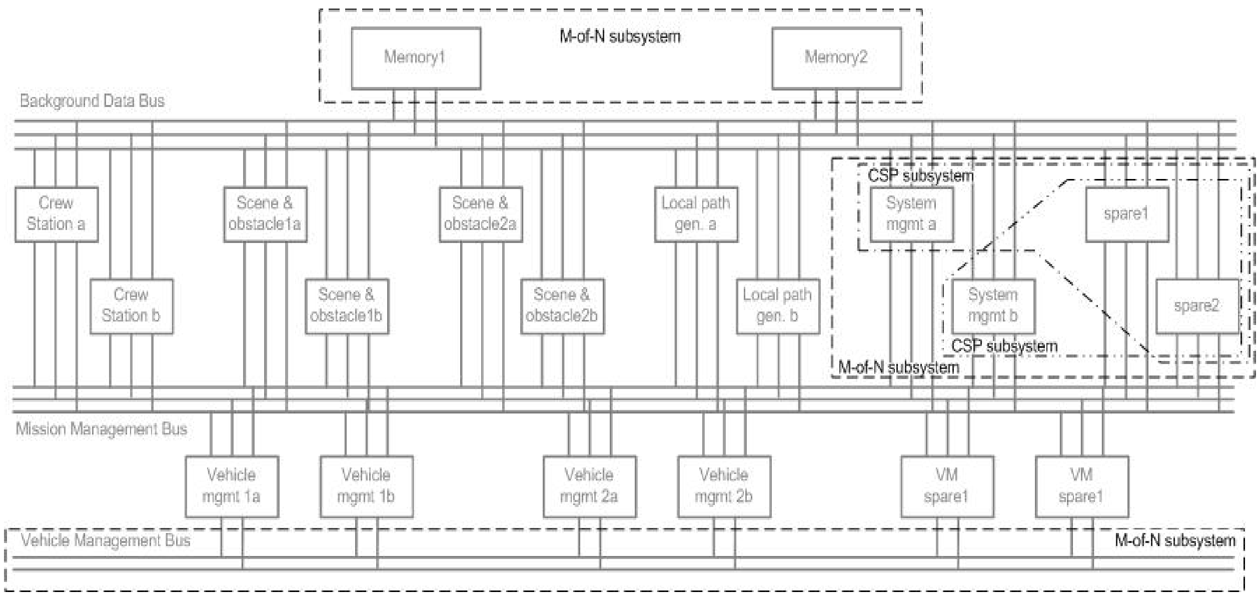


Fig. 5 Construction of subsystems to build Extended model

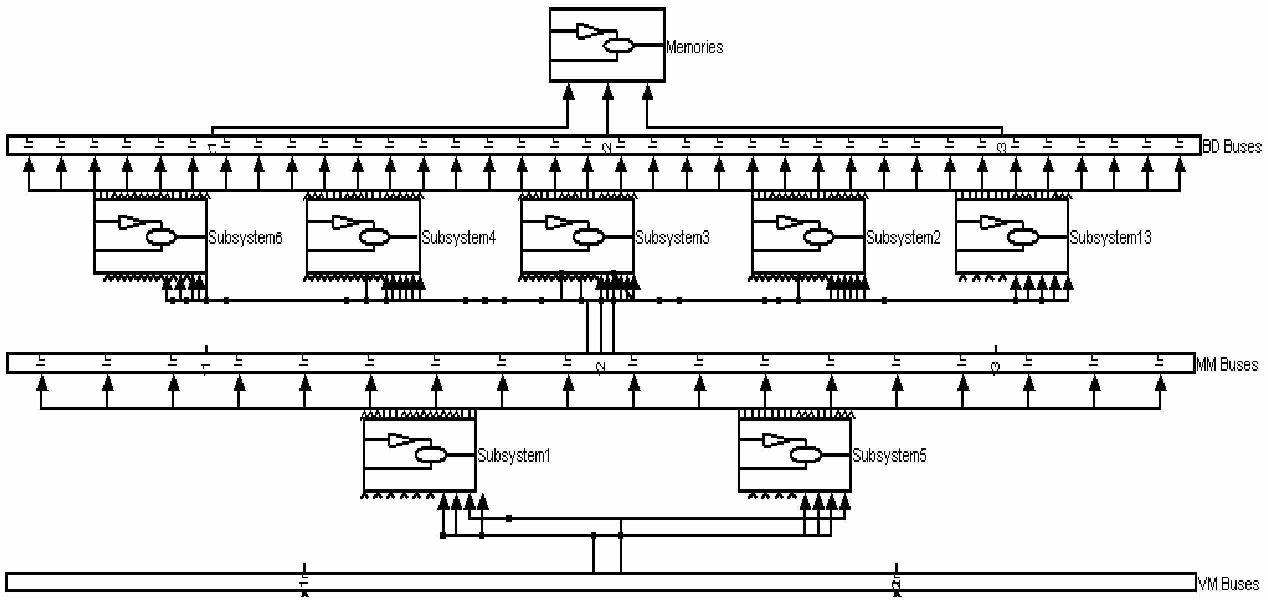


Fig. 6 Extended model of the MAS

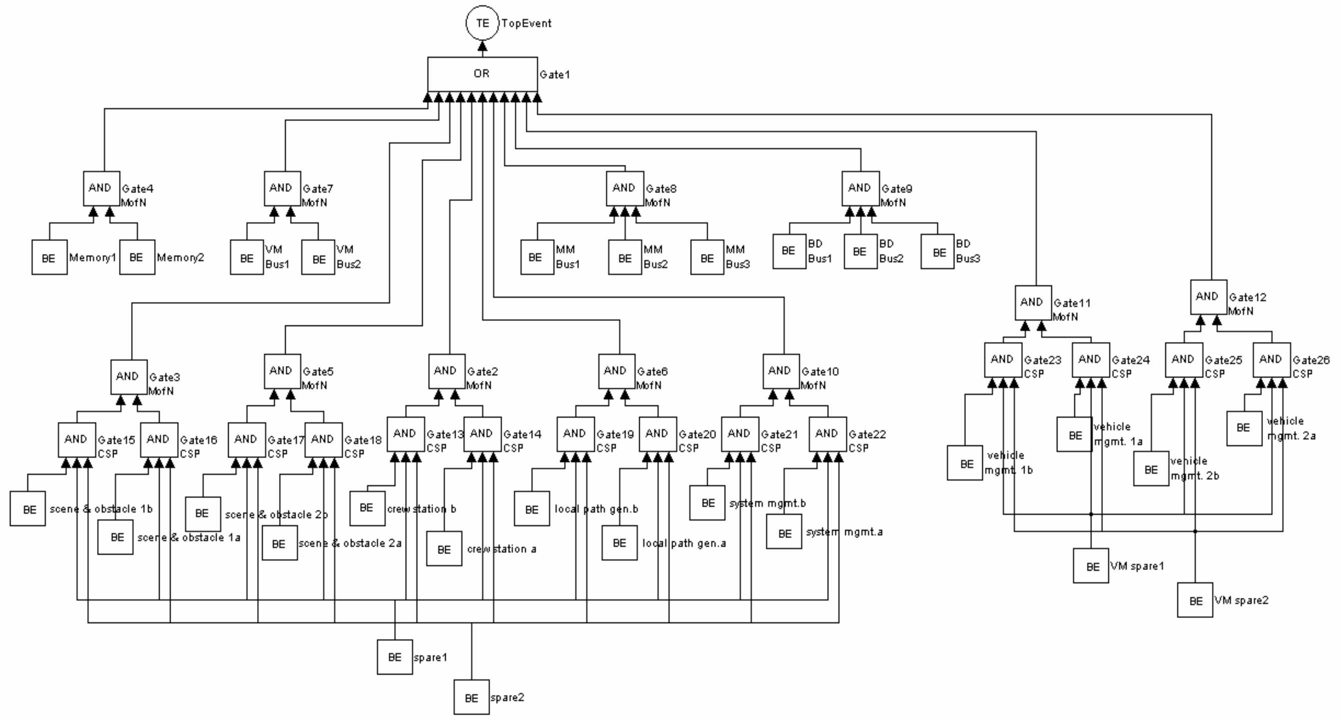


Fig. 7 The fault tree model of the MAS

IV. Advantages of the Novel Method

In this section the advantages of the purposed methodology are listed.

- Widely used MATLAB-Simulink software for modeling and simulation of engineering systems, makes it a good choice for defining system topology in our methodology.
- The use of Simulink and its powerful user-interface gives us some useful graphical capabilities for copying, resizing, scaling, moving, printing, and favorite changing by user.
- The generated fault tree can be copied and used as a sub-tree in other fault trees constructed by our tool.
- Analysis the static fault tree can perform in two ways; static and dynamic analysis.
- The results of fault tree analysis can be exported to other application softwares such as Word, Excel, Access, etc for the evaluation of the tree.
- Direct using of the system diagram, avoids the tedious working of generating digraphs, transition table, decision tables, and knowledge-based rules.
- The proposed methodology is simple and fast.

V. Conclusion

A new methodology to the automatically construction of fault trees has been proposed in this paper. System block diagram is modeled in MATLAB-Simulink and then an “Extended model”, that contains functional (or failure) and behavioral information of the system, is built manually by the user. Fault tree model of the system is then constructed automatically from this model. This methodology can analyze inferred static fault tree in two ways; static and dynamic method. The construction method was adequately explained and the potential of this approach is demonstrated by an example. We showed that the constructed fault tree from our method is identical with that of [18] which validates the correctness of our proposed methodology. Some of the advantages of our method were also mentioned.

References

[1] P. Liggesmeyer, and M. Rothfelder, “Improving System Reliability with Automatic Fault Tree Generation”, Poc. of the FTCS’28: IEEE 28th Ann. Int. Symp. on Fault Tolerant Computing Systems, Munich, 1998, pp. 90-99.

[2] P. Gaspar, and G. Szabo, “On-Line System Verification Applying an Automated Fault Tree Generation Method Integrated into Development Tools”, In the Proc. of ESREL ‘90; Ann. European Safety and Reliability Conf., Germany, 1999.

[3] E. Bourgade, N. Villatte, S. Humbert, P. Mouttapa, M. Pillière, and I. Renault, “Facilitating Risk and Dependability Analysis – A Computer Program for Automatic Fault Tree Generation: KB3”, In the Proc. of 4th Int. Conf. on Probabilistic Safety Assessment and Management, Vol. 2, New York, 1998, pp. 617-622.

[4] J. B. Fussel, “A Formal Methodology for Fault Tree Construction”, Nuclear Science and Engineering, vol. 52, 1973, 421-432.

[5] J. R. Taylor, “An Algorithm for Fault Tree Construction”, IEEE Transactions on Reliability, R-31, 1982, pp.137-146.

[6] S. A. Lapp, G. J. Powers, “Computer-Aided Synthesis of Fault-Trees”, IEEE Trans. on Reliability, R-26, 1977, pp. 2-13.

[7] A. Shafaghi, P. K. Andow, F. P. Lees, “Fault Tree Synthesis Based on Control Loop Structure”, Trans.I Chem. E, 62, 1984, pp.101.

[8] B. E. Kelly, and F. L. Lee, “The Propagation of Faults in Process Plants. Modeling of Fault Propagation”, Reliability Engineering, Vol. 16, 1986, pp.3-38.

[9] A. Bossche, “Computer-Aided Fault Tree Synthesis. System Modeling and Causal Trees.”, Reliab. Eng. Vol. 32, 1991, pp.217-241.

[10] R. C. De Vries, “An Automated Methodology for Generating a Fault Tree”, IEEE Trans. on Reliability, Vol. 39, No. 1, 1990, pp.76-86.

[11] J. D. Wang, and T. S Liu, “A Component Behavioral Model for Automatic Fault Tree Construction”, Reliability Engineering and System Safety, Vol. 42, 1993, pp.87-100.

[12] M. S. Elliot, “Computer Assisted Fault Tree Construction Using a Knowledge-Based Approach”, IEEE Transactions on Reliability, Vol. 43, 1994, pp. 112-120.

[13] Y. Wang, T. Teague, H. West, S. Mannan, “A New Algorithm for Computer-Aided Fault Tree Synthesis”, Journal of Prevention in the Process Industries, Vol. 15, 2002, pp. 265-277.

[14] K. K. Vemuri, J. B. Dugan, “Automatic Synthesis of Fault Trees for Computer-Based Systems”, IEEE Transactions on Reliability, Vol. 48, No. 4, 1999, pp. 394-402.

[15] Y. Papadopoulos, M. Maruhn, ”Model-Based Synthesis of Fault Trees from Matlab-Simulink models”, Proceeding International Conference on Dependable Systems and Networks (DSN-2001), Göteberg, Sweden, June 30th-July 4th. 2001.

[16] J. B. Dugan, S. J. Bavuso, and M. A. Boyd, “Fault Trees and Sequence Dependencies”, Proc. of the Ann. Reliability and Maintainability Symp., 1990, pp. 286-293.

[17] S. W. Behnen, W. A. Whitehouse, R. J. Farrell, “Advanced System Integration Demonstrations (ASID) System Definition”, Tech. Report; USAF Wright Aeronautical Labs. 1984.

[18] J. B. Dugan, S. J. Bavuso and M. A. Boyd, “Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems”, IEEE Transactions on Reliability, Vol. 41, No.3. 1992.