Reliability Engineering and System Safety 94 (2009) 1076–1086

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Component-based modeling of systems for automated fault tree generation

Aref Majdara *, Toshio Wakabayashi

Department of Management Science and Technology, Graduate School of Engineering, Tohoku University, Sendai 980-8579, Japan

ARTICLE INFO

Article history:
Received 20 August 2008
Received in revised form
17 December 2008
Accepted 23 December 2008
Available online 4 January 2009

Keywords:
Fault trees
Automated fault tree construction
Component-based modeling
Function table
State transition table
Flows
Trace-back algorithm

ABSTRACT

One of the challenges in the field of automated fault tree construction is to find an efficient modeling approach that can support modeling of different types of systems without ignoring any necessary details. In this paper, we are going to represent a new system of modeling approach for computer-aided fault tree generation. In this method, every system model is composed of some components and different types of flows propagating through them. Each component has a function table that describes its input–output relations. For the components having different operational states, there is also a state transition table. Each component can communicate with other components in the system only through its inputs and outputs. A trace-back algorithm is proposed that can be applied to the system model to generate the required fault trees. The system modeling approach and the fault tree construction algorithm are applied to a fire sprinkler system and the results are presented.

1. Introduction

Fault tree analysis (FTA) is now one of the most common methods for evaluating the reliability of systems, from both qualitative and quantitative points of view. Fault tree is a graphical representation of the various combinations of faults that will result in an undesired event [1].

Historically, the concept of fault tree was originated by Bell Telephone Laboratories in 1961 as a technique to perform a safety evaluation of the Minuteman Launch Control System [2]. Since then, there were significant improvements on mathematical and analytical techniques used in fault tree analysis [3].

Today, FTA is widely used in various fields of technology, mainly in aerospace, chemical, and nuclear industries, and it is finding its way into many other fields such as robotics, rail transportation, and car industries. However, manual fault tree construction is a time-consuming task, which needs much effort, especially for large-scale systems. Also, human mistakes may appear in different steps of manual fault tree construction. Therefore many attempts have been made to automate the fault tree construction procedure [4–23]. Automated fault tree generation is faster and easier than the manual approach, and is obviously more suitable for being used in design phase, where the designers may want to try lots of different configurations to find the most reliable one.

First step in automated fault tree synthesis is to model the system in an appropriate way for being used in algorithmic procedures of automated fault tree construction. One of the challenges in this field is to find an efficient system modeling approach, which can support modeling of different types of systems in a straightforward and easy to understand way, without ignoring even the smallest necessary details about the components specifications and system operation.

Various modeling approaches are utilized by different researchers. Some of the most commonly used approaches are digraphs [5–10,24], decision tables [11,12], and state diagrams [13–15]. Kumamoto and Henley proposed a semantic network modeling approach [16]. Papadopoulos et al. used Matlab–Simulink models for model-based synthesis of fault trees [17,18]. Some other studies were based on utilization of other special techniques like text mining [19] and case-based reasoning [20] for automated generation of fault trees. Some researches focused on a special type of systems, like automated fault tree generation for electrical circuits by De Vries [21].

Knowledge-based approaches have been the basis of some powerful computerized fault tree generation tools like KB3 [25], STARS [26], and PC-FTA [27]. Latif-Shabgahi has compared the different aspects of selected knowledge-based fault tree construction tools in his paper [28]. Among these tools, KB3 is now a commercially available tool with useful features. It is a knowledge-based tool that automatically generates fault trees

* Corresponding author. Tel./fax: +81 22 795 4838.
E-mail address: majdara.aref@most.tohoku.ac.jp (A. Majdara).

0951-8320/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2008.12.003

from the description of a system. The knowledge bases are written with the FIGARO language [29] and are used for system modeling in a user-friendly graphical interface. Having the system model, fault trees are generated using backward chaining rules.

In this paper, we are going to represent a component-based system modeling method for computer-aided fault tree generation. The main purpose is to develop a modeling approach, which is able to model a wide range of systems containing various types of components. We have tried to combine the useful ideas from different modeling approaches like digraphs, decision tables, state diagrams, etc. into one unit structure. Because each of these methods has special capabilities in modeling specific kinds of systems, while any of them has its own weaknesses in some other features.

Section 2 gives a general overview of the proposed approach. Basic concepts and elements of a system model, like components, flows, etc. are introduced in Section 3. In Section 4, we will see how to use the basic concepts introduced in Section 3 to model a system; and we will try to make it clearer by applying the approach to a sample system in Section 5. Section 6 explains the other part of the automated fault tree generation procedure, which is fault tree synthesis using the system model, and a trace-back algorithm. The fault tree for the sample system is presented in Section 7. Some complementary issues about the features of this approach will be discussed in Section 8. And finally, conclusions presented in Section 9.

2. General description of the approach

This modeling method is a component-based approach, meaning that we model any system as a set of components connected to each other. Different kinds of materials, commands, energy, or information can be transferred as flows through the connections between the components. In general, components are the building blocks of the systems and flows move among the components in the system. Each component has some input and output connectors and its operational specifications are described in its function table and state transition table.

We model systems by using some of previously defined components, connecting them to each other in an appropriate way, and making the required settings.

After the system is modeled, the top event can be defined by determining specific values or conditions for some of the parameters of the system. Then, a fault tree synthesis algorithm starts its job from the point of occurrence of the top event, and traces back all different possible paths that can lead into that top event. The trace-back continues until it reaches a system boundary, or a point for which more progress is either impossible or unnecessary.

3. Basic concepts in a system model

3.1. Components

We consider any of the physical elements of a system as a *component*. This can include all the electrical and mechanical devices, etc. They can be either active components like pumps and heaters, or passive components like pipes and wires. Also, human operators are modeled as special components.

Furthermore, there are some virtual components like junctions and extensions, defined for modeling purposes. These are not actual parts of the systems, and just help us to implement some special concepts of the system. These will be discussed in detail later in this paper.

Each component is symbolized as a simple box with a label, and some arrows pointing inward or outward, as its input and output paths, respectively. Fig. 1 shows the component model for a simple pump with 2 inputs and 1 input. This pump has an input for electricity and one for the liquid flow. The only output carries the liquid out of the component.

Each output (input) of a component can be connected to only one input (output) of another component. And this is how we model the hardware configuration of the system. However, a component is not supposed to be a simple box with some inputs and outputs. The most important part of a component model is the way it functions, i.e. how it processes the inputs to produce the outputs. The functional descriptions of the components are explained in Section 3.2.

3.2. Function tables

The input–output relations of any component are described in a *function table*. Using a function table, we will be able to determine the component's output for each combination of input values, component's functionality condition, and other parameters. A sample function table for a pump is shown in Table 1. The pump has one power input and a fluid input. It has a single output for the fluid.

The output is a function of the inputs, and the functionality condition. The two inputs are coming from other components of the system, but the functionality condition is an internal parameter of the component. In case of our sample pump, the output is always zero, except for the situation described in the last row of the table.

3.2.1. Functionality condition

A system fails when some of its components are in a failed state. *Functionality condition* is an internal parameter of a component, describing its ability or disability to perform the desired function. In the simplest form, as for the pump in Table 1, the component can be either in working or failed state. However, in general there can be more than one *failure mode* for a component. For example, a simple valve may fail to open, fail to close, or have some leakage. During creating fault trees, the failure modes of components usually appear as the basic events, terminating the branches of the trees.

Any parameter appearing in a function table has its own range of variation. These ranges determine the number of the rows for

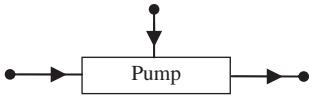

Fig. 1. Component model for a pump with 2 inputs and 1 output.

Table 1
Sample function table for a pump.

Input 1 (Power)	Input 2 (Fluid)	Functionality condition	Output
0	0	Failed	0
0	0	Working	0
0	1	Failed	0
0	1	Working	0
1	0	Failed	0
1	0	Working	0
1	1	Failed	0
1	1	Working	1