The Algorithm of Automated Development of Fault Trees for Safety Exploitation Assessment of Complex Technical Systems

L.Ozirkovskyy, A. Mashchak, O. Shkiliuk, S. Volochiy

Abstract — this article presents an algorithm, which allows to automate the developing of fault trees for the safety exploitation assessment of complex technical systems. As result, this algorithm produces logical and graphical representations of fault tree. Obtained fault tree can be used for the exploitation safety assessment. Presented algorithm is a further step in the development of the technology of modeling the discrete-continuous stochastic systems, based on so-called structural-automaton models.

Keywords — safety, reliability, safety engineering, fault tree analysis, complex technical system, algorithms, structural-automaton model.

I. Introduction

For the effective functioning of the complex technical system (henceforth named CTS), it's extremely important to provide the required levels of reliability and safety. The failure of CTS can have devastating consequences that can lead to significant damage, including the loss of lives. Therefore, problems of safety, reliability and failure consequences assessment, during the design stage, are very important. The assessment of the reliability and exploitation safety of CTS is carried out by modeling, which is used for determining reliability and exploitation safety indicators.

There are several approaches for reliability and exploitation safety assessment, namely: simulation modeling (hereafter SM), stochastic simulation and state space modeling. Stochastic simulation has been indicated as being the most commonly used approach [1]. Important feature of stochastic simulation is that algorithms for developing models are well-formalized, so they are simple to program, and there are quite a lot of software based on that algorithms.

The most commonly used method of stochastic simulation of CTS involves the fault tree (henceforth named FT) development [2]. FT is a type of structure flowchart, which is used for graphical representation of events (determined with deductive method) that can lead to catastrophic system failure [3]. FT analysis can be used to obtain minimal cut sets (hereafter MCS). MCS is a minimal combination of events that leads to system failure. If any event is removed from the MCS, the remaining events collectively are not able to cause the system failure. MCS, obtained via FT analysis, can also include the probability of MCS (probability of all the events from the cut set occur in the same time).

II. Review of existing approaches of fault tree development

In modern literature, there are many works regarding FT development and analysis. Articles [2-5] describe general principles and features of FT development. Articles [6-12] present approaches of FTs development and analysis, with their further application to specific systems.

In [6] authors present an approach for dynamic FTs development and analysis. A positive feature of this approach is that it considers characteristics such as time dependencies and repeated events. After dynamic FT is developed, authors use Monte-Carlo SM method for

L. Ozirkovskyy, Lviv Polytechnic National University, Lviv, Ukraine (e-mail: lozirkovsky@lp.edu.ua)
A. Mashchak, Lviv Polytechnic National University, Lviv, Ukraine (e-mail: maschak@lp.edu.ua)
O Shkiliuk, Lviv Polytechnic National University, Lviv, Ukraine (e-mail: shkiliuk@lp.edu.ua)
S. Volochiy, Lviv Polytechnic National University, Lviv, Ukraine (e-mail: volochiy.s@gmail.com)

further analysis of the system. It should be noted, however, that the usage of SM for dynamic FT analysis significantly increases the time costs in comparison with static FT analysis.

Article [7] presents an approach for the development of the static FT with time dependencies. Approach is described with an example of development and analysis of the FT model of the telecommunication network.

Paper [8] presents a method for FTs development and analysis, based on decomposition. In this method, FT is decomposed into simpler disjoint FTs, which are “analyzable”. The results from the analysis of all simpler FTs are re-combined to obtain the results for the original FT. Using the presented method, it’s possible to obtain minimal cut sets and the probability of the failure of the system. However, it should be noted that, because of decomposing and re-combining the FT, the determined value of the failure of the system will contain an error.

Paper [9] focuses on a sub-class of dynamic FT, which are called Priority Dynamic FTs. Described Priority Dynamic FTs use so called Priority Dynamic Gates and Repeated Events to consider dependencies between events, including failure events. However, further analysis of such FTs requires the use of SM methods or state space method.

Paper [10] investigates the problem of mutual influence of basic events in Noncoherent FTs. Authors present an approach of the development of FTs, which considers changing the failure probability at certain conditions. The results are demonstrated with an example, where authors develop the model of emergency shutdown system in nuclear power plant. This approach is complicated, involves significant effort and engineer time for FT design and analysis.

Article [11] compares two methods of analysis – FT analysis and failure mode and effects analysis (FMEA). Authors present a comparative analysis of these two tools, showing the contribution of each one for the implementation of a structured predictive maintenance planning for the hydraulic turbines.

Work [12] presents method of the analysis of the reliability of telecommunication systems. Presented method is based on FT analysis. As a part of this method, author proposes principles of the development of models of telecommunication systems together with models of their technical exploitation systems. Such models can be used for ensuring the reliability of systems and for determining the importance of failures of the system. With the development of the technical exploitation models, author emphasize the importance of considering not only the structure of the technical system, but also its functional behavior.

Authors of an article [13] propose the method of FT development, which is based on load-sharing redundancy. The main disadvantage of the proposed method is that is difficult to use and it requires large amount of time for model development.

The overview of modern methods of FT development and analysis shows us, that in most cases, "manual work" is required. Also, most methods consider that all basic events in the system are independent (FT is static). On the other hand, use of dynamic FTs, which allows to consider interconnection between events, requires involving complicated methods for further analysis (SM, states space method), that entails significant costs of time and effort.

Besides that, known methods of FTs development and analysis are not able to consider reliability and functional behavior of the CTS in one model. The maintenance and limitations of repairs number, the tools of control and diagnosis, the reliability of software are not considered also.

Therefore, there is an important problem of development of an approach, which would allow to consider aforementioned features of CTS. It’s also important for such approach to be well-formalized so it would be possible to automate the process of obtaining FT and MCS.

III. The proposed approach for the automation of the development of the fault tree of the complex technical system

The proposed approach for the automation of the development of FTs is based on an existing method of determining MCS directly from the state-transition model of the object of investigation [14]. The other basis of the proposed approach is the state space method [15], in particular its improved version [16, 17]. This improved version of the state space method is based on the well-formalized method of the development of the state-transition models, which allows to automate the development of mathematical models of the behavior of CTS, considered as discrete-continuous stochastic systems.

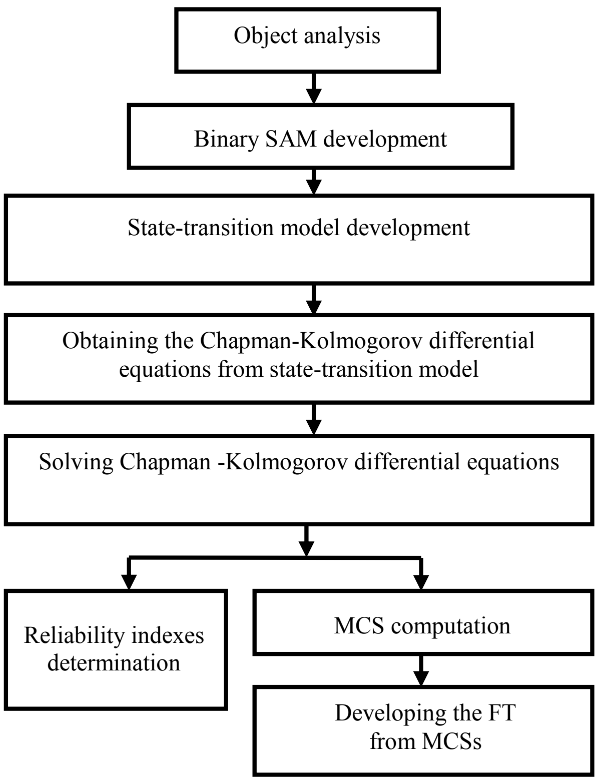
Object analysis
Binary SAM development
State-transition model development
Obtaining the Chapman-Kolmogorov differential equations from state-transition model
Solving Chapman -Kolmogorov differential equations
Reliability indexes determination
MCS computation
Developing the FT from MCSs

Fig. 1. Fault tree development methodology

The automation of the development of mathematical models of the behavior is based on so-called "structural-automaton" model (SAM) of the object of investigation. SAM of CTS is a formalized representation of its structure and behavior. SAM consists of three parts. First part of it is a set of parameters, considered in the model. These parameters are listed in a state vector. Second part – list of events that cause change in the system's state. These events are called "basic events". Third part of SAM – set of expressions that describe the rules of state vector modification, caused by basic events. There is an algorithm to develop state-transition model from SAM. However, the level of detail of representation of the object of investigation in obtained model is insufficient for developing the fault tree (FT). For solving this problem, the method of development of the SAM has been improved so we could develop so called "binary SAM".

The binary SAM is the SAM of the object of investigation, in which every considered component of system's structure is represented by individual state vector component which can take only zero/one values. The point of developing the binary SAM, is that, unlike to ordinary SAM [15, 16], in binary SAM it is possible to describe the structure and behavior of CTS without coupling states of its structure components. Thereby, it is possible to obtain the description of specific failure state which will include states of inoperative components with required level of details.

Using ASNA software tool [17], it is possible to generate state-transition model from binary SAM. State-transition model, generated from binary SAM, will have a sufficient level of details of the object of investigation, required for FT development. For generated state-transitions model, ASNA software tool builds the analytical model of CTS in form of the system of linear differential Chapman - Kolmogorov equations. The solution of these equations gives the probability distribution of the CTS being in all possible states. Using the proposed procedure of filtering failure states, MCSs can be obtained [14]. Quantitative indicators of MCSs can be obtained by summing the probabilities of the system being in corresponding states.

Described technique of FT development is presented on Figure 1.

IV. The automation of the development of the Fault tree of complex technical system

In proposed approach, the MCS array is used as an input data for FT development. As a result of procedures of the proposed approach, the logical function will be obtained. With this logical function, the graphical representation of FT can be built.

Automated FT development is performed in two stages. On the first stage FT logical function is obtained by sequential sorting of MCS array. Based on obtained FT logical function the FT graphical representation is built on the second stage.

It should be noted that obtained FT is static but probabilistic indexes of MCS consider dynamic processes which occur in CTS.

According to presented technique an algorithm which builds FT logical function is developed and is implemented in software prototype.

A. Algorithm of obtaining the fault tree logical function

In presented algorithm of the first stage the sequential record of FT logical function is performed.

Abbreviations which are used in description of algorithm for automated obtaining of FT logical function:

n – MCS serial number pointer.

i – serial number of the SV component pointer.

j – zero value of SV component counter; this counter keeps the number of SV components whose values are zero.

CSVC – constant of SV components number.

ZVC$_n$ – zero value counter; this counter keeps the number of SV components whose values are zero for MCS with serial number **n**.

CMCS – constant of MCS number.

SV$_n$[i] – value of the **i**-th SV component for MCS with serial number **n.**

LFF – logical function form.

Step 1. Take the first MCS. In MCS serial number pointer **n** the **1** is recorded and the number of SV components, whose values are zero for the first MCS, is recorded in zero value counter **ZVC$_1$** of first MCS. In zero value SV component counter **j** the number of SV components, whose values are zero, is recorded: **j = ZVC$_1$**.

Step 2. The first SV component of the first MCS is taken: **i = 1.** In **LFF** a "(" symbol is written.

Step 3. On this step the value of **i**-th SV component of the MCS with serial number **n** is needed to be determined (in first cycle of algorithm the first SV component of the first MCS). So there such variants can take place:

If value of the **i**-th SV component of the MCS with serial number **n** is equal to zero (**SV$_n$[i]** = 0), then in logical function form **(LFF)** the value of serial number of the SV component pointer **i** is written V**i**. Written zero value SV component counter **j** is decremented by one: **j** = **j-1** and after this **j** is checked by inequality **j < 0**.

If **j > 0** it means that in MCS with serial number **n** there are SV components whose values are zero. So in **LFF** a "·" symbol is written, then go to step 4.

If **j** = 0 it means that in MCS with serial number **n** there are no more SV components whose values are zero, then go to step 4.

If value of the **i**-th SV component of the MCS with serial number **n** is not equal to zero (**SV$_n$[i]** > 0), then go to step 4.

Step 4. Serial number of the SV component pointer **i** of the MCS with serial number **n** is incremented by one: **i = i+1.** After incrementation pointer **i** is checked by comparison with **CSVC.**

If **i ≤ CSVC** it means that the pointer **i** is still in array of SV components.

If **i > CSVC** it means that the pointer **i** got out last SV component:

If **i ≤ CSVC**, then go to step 3.

If **i > CSVC**, then go to step 5.

Step 5. Minimal cut sets serial number pointer **n** is incremented by one: **n = n+1**. Now it is checked if the pointer **n** got out MCS array by its comparison with **CMCS**:

If **n ≤ CMCS**, then in **LFF** the ") +" symbols are written and go to step 2.

If **n > CMCS**, then in **LFF** the ");" symbols are written. In this stage the procedure of writing FT logical function is finished.

B. *Algorithm of building the fault tree graphical representation*

Step 1. First stage is the building of the lowest levels of FT. For this all elements, which are in brackets and between which are multiply symbols, are combined by logical elements AND. All groups of elements, between whose elements there are multiply symbols, have only one output. In result the lowest level of fault tree is built. After that it should go to step 2.

Step 2. In this stage next levels and the highest event of fault tree are formed. For this all outputs from lower levels are combined by logical elements OR according to elements combining rule.

Elements combining rule – logical elements AND or OR can combine only two outputs from lower level or two elements in step 1.

Using this technique, the FT is built.

V. Fault tree building example of fault tolerant hardware/software system

Fault-tolerant hardware/software system (FTH/SS) is object which consists of four main modules: input receiving module, operating module, data processing module and executing output module. All these modules perform an objective function of FTH/SS. Input receiving module and data processing module are reserved. Failure of FTH/SS can occur in result of hardware or software failure. Block diagram of FTH/SS is shown in Figure 2.

According to presented in [14] technique the MCS array was obtained for FTH/SS and it is presented in Table 1.

Input data:

CSVC – assigns to 7 – there are 7 SV components.

CMCS – assigns to 4 – there are 4 MCS.

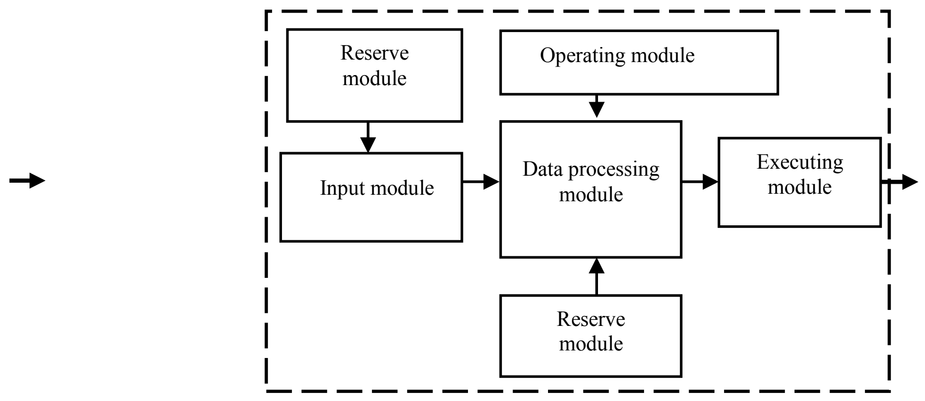

Fig.2. Block diagram of fault-tolerant hardware/software system.

TABLE I
MINIMAL CUT SETS ARRAY

Serial numbers of MCS	State vector components
1	V1=1; V2=1; V3=1; V4=1; V5=1; V6=1; V7=0
2	V1=1; V2=1; V3=1; V4=1; V5=0; V6=0; V7=1
3	V1=1; V2=1; V3=0; V4=0; V5=1; V6=1; V7=1
4	V1=0; V2=0; V3=1; V4=1; V5=1; V6=1; V7=1

A. Algorithm of obtaining the fault tree logical function

Step 1. Take the **1**-st MCS and the **1**-st SV component: **n = 1**; **j** = **1**;

Step 2. **i** pointer assigns to 1: **i = 1;**

In **LFF** write a "(" symbol and go to step 3.

Step 3. Take value of the **1**-st SV component of the **1**-st MCS. As **(SV_1[1] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 1+1=2.** As **2 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **2**-nd SV component of the **1**-st MCS. As **(SV_1[2] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 2+1=3.** As **3 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **3**-rd SV component of the **1**-st MCS. As **(SV_1[3] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 3+1 = 4.** As **4 < 7(i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **4**-th SV component of the **1**-st MCS. As **(SV_1[4] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 4+1 = 5.** As **5 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **5**-th SV component of the **1**-st MCS. As **(SV_1[5] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 5+1 = 6.** As **6 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **6**-th SV component of the **1**-st MCS. As **(SV_1[6] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 6+1 = 7.** As **7 = 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **7**-th SV component of the **1**-st MCS. **SV_1[7] = 0**.

In **LFF** write V**i** – **V**7.

j = j-1; j = 1-1 = 0. As **j = 0**, so go to step 4.

LFF: (V7

Step 4. **i = i+1; i = 7+1 = 8.** As **8 > 7 (i > CSVC)**, so go to step 5.

Step 5. **n = n+1; n = 1+1=2.** As **2 < 4 (n ≤ CMCS)**, so in **LFF** write ") +" symbols and go to step 2.

LFF: (V7) +

Step 2. **i = 1**; **j** = **2**. In **LFF** write "(" symbol and go to step 3:

LFF: (V7) + (

Step 3. Take value of the **1**-st SV component of the **2**-nd MCS. As **(SV_2[1] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 1+1 = 2.** As **2 < 7 (i ≤ CSVC)**, sogo to step 3.

Step 3. Take value of the **2**-nd SV component of the **2**-nd MCS. As **(SV_2[2] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 2+1 = 3;** As **3 < 7 (i ≤ CSVC)**, so go to 3.

Step 3. Take value of the **3**-rd SV component of the **2**-nd MCS. As **(SV_2[3] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 3+1 = 4. 4 < 7 (i ≤ CSVC)**, go to step 3.

Step 3. Take value of the **4**-th SV component of the **2**-nd MCS. As **(SV_2[4] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 4+1 = 5.** As **5 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **5**-th SV component of the **2**-nd MCS. **SV_2[5] = 0**.

In **LFF** write V**i** – **V5:**

j = j -1; j = 2-1 = 1. As **j > 0**, so write a "·" symbol and go to step 4.

LFF: (V7) + (V5 ·

Step 4. **i = i+1; i = 5+1 = 6.** As **6 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **6**-th SV component of the **2**-nd MCS. **SV_2[6] = 0.**

In **LFF** write V**i** – **V6:**

LFF: (V7) + (V5 · V6

j = j-1; j = 1-1=0; As **j = 0**, so go to step 4.

Step 4. **i = i+1; i = 6+1 = 7.** As **7 = 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **7**-th SV component of the **2**-nd MCS. As **(SV_2[7] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 7+1 = 8.** As **i > CSVC**, so go to step 5.

Step 5. **n = n+1; n = 2+1 = 3;** As **3≤ 4** (**n < CMCS)**, so in **LFF** write ") +" symbols and go to step 2.

LFF: (V7) + (V5 · V6) +

Step 2. **i = 1; j = 2.**

In **LFF** write a "(" symbol and go to step 3:

LFF: (V7) + (V5· V6) + (

Step 3. Take value of the **1**-st SV component of the **3**-rd MCS. As **(SV_3[1] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 1+1 = 2.** As **2 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **2**-nd SV component of the **3**-rd MCS. As **(SV_3[2] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 2+1 = 3.** As **3 < 7(i ≤ CSVC)**, so go to step 3.

Step 3. **SV_3[3] = 0.**

In **LFF** write V**i** – **V3:**

LFF: (V7) + (V5· V6) + (V3

j = j-1; j = 2-1=1. As **j > 0** so write a "·" symbol and go to step 4.

LFF: (V7) + (V5· V6) + (V3 ·

Step 4. **i = i+1; i = 3+1=4.** As **4 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. **SV_3[4] = 0;**

In **LFF** write V**i** – **V4:**

LFF: (V7) + (V5· V6) + (V3 · V4

j = j -1; j = 2 -1=1. As **j = 0,** so go to step 4.

Step 4. **i = i+1; i = 4+1=5;** As **5 < 7 (i ≤ CSVC),** so go to step 3.

Step 3. Take value of the **5**-th SV component of the **3**-rd MCS. As **(SV_3[5] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 5+1=6.** As **6 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **6**-th SV component of the **3**-rd MCS. As **(SV_3[6] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 6+1=7;** As **7 = 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **7**-th SV component of the **3**-rd MCS**.** As **(SV_3[7] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 7+1=8.** As **8 > 7 (i > CSVC)**, so go to step 5.

Step 5. **n = n+1; n = 3+1= 4.** As **4 = 4 (n ≤ CMCS)**, so in **LFF** write ") +" symbols and go to step 2.

LFF: (V7) + (V5 · V6) + (V3 · V4) +

Step 2. **i = 1; j = 2.**

In **LFF** write a "(" symbol and go to step 3:

LFF: (V7) + (V5 · V6) + (V3 · V4) + (

Step 3. Take value of the **1**-st SV component of the **4**-th MCS**. SV_4[1] = 0**.

In **LFF** write V**i** – V1.

j = j-1; j = 2-1=1. As **j > 0,** so write a "·" symbol and go to step 4.

LFF: (V7) + (V5 · V6) + (V3 · V4) + (V1 ·

Step 4. **i = i+1; i = 1+1=2.** As **2 < 7 (i ≤ CSVC),** so go to step 3.

Step 3. Take value of the **2**-nd SV component of **4**-th MCS. **SV_4[2] = 0.**

In **LFF** write V**i** – V2.

LFF: (V7) + (V5 · V6) + (V3 · V4) + (V1 · V2

j = j-1; j = 1-1=0. As **j = 0**, so go to step 4.

Step 4. **i = i+1; i = 2+1=3.** As **3 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **3**-rd SV component of **4**-th MCS. As **(SV_4[3] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 3+1=4.** As **4 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **4**-th SV component of **4**-th MCS. As **(SV_4[4] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 4+1=5.** As **5 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **5**-th SV component of the **4**-th MCS. As **(SV_4[5] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 5+1=6. 6 < 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **6**-th SV component of the **4**-th MCS. **(SV_4[6] = 1)** > 0.

Step 4. **i = i+1; i = 6+1=7.** As **7 = 7 (i ≤ CSVC)**, so go to step 3.

Step 3. Take value of the **7**-th SV component of the **4**-th MCS. As **(SV_4[7] = 1)** > 0, so go to step 4.

Step 4. **i = i+1; i = 7+1=8.** As **8 > 7 (i > CSVC)**, so go to step 5.

Step 5. **n = n+1; n = 4+1=5;** As **5 > 4 (n > CMCS)**, so in **LFF** write ");" symbols.

LFF: (V7) + (V5 · V6) + (V3 · V4) + (V1 · V2);

Procedure of writing the FT logical representation is finished.

According to algorithm of building the FT graphical representation and basing on obtained LFF the FT of fault-tolerant H/S system was built and is presented in Figure 3.

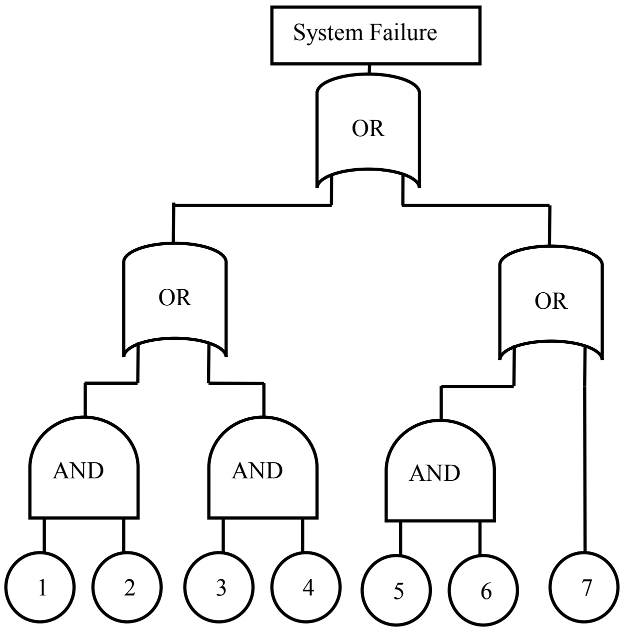

Fig.3. Fault tree of fault tolerant H/S

VI. Conclusions

The developed software tool that is based on presented approach significantly reduces time required for the analysis of reliability and exploitation safety of complex technical systems. Especially this reduction is noticeable when it is necessary to consider multiple configurations of investigated system, because the development of the fault tree for each implementation variant is automated.

Presented technique of automated fault tree development and analysis provides high certainty of recommendations regarding increasing reliability and exploitation safety of complex technical systems.

References

[1] Mozhaev A. S. Theoretical Fundamentals, Experience of Application and Direction of Development of General Logical-probabilistic Method of Software Complex "ARBITR" for Reliability Modeling, Durability, Safety and System risk / A. S. Mozhaev // Proceedings of JSC "SPIK SZMA", St. Petersburg.- 2010.- 11 p. [in Russian]

[2] Shcherbovskykh S.V. Mathematical Models and Methods for Dependability Characteristic Determination of Repairable Multi-terminal Load-shared Systems / S.V. Shcherbovskykh. – Lviv: Lviv Polytechnic National University Publ., 2012. – 296 p. [in Ukrainian]

[3] Henley E. J. Reliability Engineering and Risk Assessment / E. J. Henley, H. Kumamoto // Prentice-Hall, - 1981. - 568 p.

[4] Kececioglu D. Reliability Engineering Handbook, Vol. 2 / D. Kececioglu, - Prentice Hall Inc.: New Jersey, 1991. - 541 p.

[5] Liggesmeyer P. Fault Tree Analysis, Current Research Issues, Tutorial / P. Liggesmeyer, B. Kaiser // Proceed SAFECOMP 2004, Potsdam 2004.

[6] Chiacchio F. Dynamic Fault Trees Resolution: a Conscious Trade-Off Between Analytical and Simulative Approaches / F. Chiacchio, L.Compagno, D. D’Urso, G. Manno, N.Trapani // Reliability Engineering & System Safety. - 2011. - Vol. 96, № 11. - P. 1515-1526.

[7] Skrobanek P. Analysis of Timing Requirements for Intrusion Detection and Prevention Using Fault Tree with Time Dependencies / P. Skrobanek, M.Woda // In: Skrobanek P, editor. Intrusion detection systems. InTech.- 2011.- P. 307-324.

[8] Contini S. Analysis of Large Fault Trees Based on Functional Decomposition / Sergio Contini, Vaidas Matuzas // Reliability Engineering & System Safety. - 2011. - Vol.96, № 3. - P. 383-390.

[9] Merle G. Probabilistic Algebraic Analysis of Fault Trees With Priority Dynamic Gates and Repeated Events / G. Merle, J.-M. Roussel, J.-J. Lesage, A. Bobbio // Reliability, IEEE Trans. on. - 2010. - Vol. 59, № 1. - P. 250-261.

[10] Lixuan Lu. Joint Failure Importance for Noncoherent Fault Trees / Lu Lixuan, Jin Jiang // Reliability, IEEE Trans. on. - 2007. - Vol. 56, № 3. - P. 435-443.

[11] Rodrigo de Queiroz Souza FMEA and FTA Analysis For Application of The Reliability Centered Maintenance Methodology: Case Study on Hydraulic Turbines / Rodrigo de Queiroz Souza, Alberto José Álvares // ABCM Symposium Series in Mechatronics - Vol. 3 - P. 803-812.

[12] Maevskyy L. S. Methods of Reliability Providing of Information - Telecommunication Systems in Different Life Cycle Stages / L. S. Maevskyy. - St. Petersburg: Barzilovich Z. P. Publ., 1999, 112 p. [in Russian]

[13] Mandziy B. Mathematical Model for Failure Cause Analysis of Electrical Systems with Load-sharing Redundancy of Component / B. Mandziy, O. Lozynsky, S. Shcherbovskykh, // Przegląd Elektrotechniczny, 89 (2013), No 11, pp. 244-247.

[14] Volochiy, B. The new method of building a safety model for quantitative risk assessment of complex technical systems for critical application /Volochiy, B.,Mandziy, B.,Ozirkovskyy, L.// Communications in Computer and Information Science. Volume 594. Information and Communication Technologies in Education, Research, and Industrial Applications, Publisher: Springer International Publishing, 2016, pp.56-70

[15] Tarakanov K. V. Analytical Methods of Systems Research / K. V. Tarakanov, L. A. Ovcharov, A. N. Tyryshkin,- M .: Sov. Radio, 1974, 240 p. [in Russian]

[16] Volochiy B. Yu. Modeling Technology of Behavior Algorithms of Information Systems / Volochiy B.Yu. – Lviv: Lviv Polytechnic National University Publ., 2004, 220 p. [in Ukrainian]

[17] Bobalo Yu. Mathematical Models and Methods of Analysis of Radioelectronic, Electromechanic and Software Systems / Yu. Bobalo, B. Volochiy, O. Lozynskyy, B. Mandziy, L. Ozirkovskyy, D. Fedasyuk, S. Shcherbovskykh , V. Yakovyna. – Lviv: Lviv Polytechnic National University Publ., 2013, 300 p. [in Ukrainian]