Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

Identifying critical architectural components with spectral analysis of fault trees

Tolga Ayav [a,*], Hasan Sözer [b]

[a] Izmir Institute of Technology, İzmir, Turkey
[b] Ozyegin University, İstanbul, Turkey

ARTICLE INFO

Article history:
Received 10 December 2015
Received in revised form 22 June 2016
Accepted 30 June 2016
Available online xxx

Keywords:
Hardware/software architecture evaluation
Reliability analysis
Fault trees
Fourier analysis
Sensitivity analysis
Importance analysis

ABSTRACT

We increasingly rely on software-intensive embedded systems. Increasing size and complexity of these hardware/software systems makes it necessary to evaluate reliability at the system architecture level. One aspect of this evaluation is sensitivity analysis, which aims at identifying critical components of the architecture. These are the components of which unreliability contributes the most to the unreliability of the system. In this paper, we propose a novel approach for sensitivity analysis based on spectral analysis of fault trees. We show that measures obtained with our approach are both consistent and complementary with respect to the recognized metrics in the literature.

1. Introduction

We increasingly rely on software-intensive systems such as modern embedded systems employed in telecommunication, consumer electronics, automotive, avionics and health application domains. Software plays a central role in defining the functionality and the quality for these systems. As a result, both hardware and software faults constitute a threat for system reliability. This threat gets amplified as systems continue to grow in size and complexity.

For a long period, software reliability has been basically addressed at the source code level. However, the increasing size and complexity required a special focus on higher abstraction levels as well. In particular, early reliability evaluation at the software architecture design level became essential [1,2]. Software architecture represents the gross level structure of the system, consisting of a set of components, connectors and configurations [3,4]. This structure has a significant impact on the reliability of the system [5]. Hence, it is important to evaluate software architecture with respect to reliability risks [6]. By this way, the quality of the system can be assessed early to avoid costly redesigns and reimplementations.

In the case of software-intensive embedded systems, both hardware and software faults have to be taken into account. To analyze the propagation and interaction of these faults, system level abstract models have been developed. These models include state/path-based models [5,7,8] and (dynamic) fault trees [9,10]. In this work, we employ fault tree models, which depict logical interrelationships among faults that cause a system failure. They have been integrated as part of AADL (Architecture Analysis and Design Language) [11]. There also exist tools for synthesizing them automatically based on UML models [12]. Fault trees can be used for estimating the reliability of the overall system based on individual component failures. Another goal is to estimate the sensitivity of system reliability with respect to reliabilities of system components [5,13,14]. This goal is achieved with so-called *sensitivity analysis* [15] or *importance analysis* [16,17] to identify critical components [18]. These are the components of which unreliability contributes the most to the unreliability of the system.

An established measure for sensitivity/importance was introduced by Birnbaum [19], which is basically defined as the partial derivative of the system reliability with respect to the corresponding component reliability. Hereby, the system reliability is defined as a function of reliabilities of the involved components. There have also been other measures introduced for assessing component importance; however, it was later observed that they provide counterintuitive or inconsistent results [16,20].

In this paper, we propose a novel approach for sensitivity analysis. The approach is based on the spectral analysis of Boolean functions. Spectral (or Fourier) analysis is widely used in mathematics and engineering for decomposing a signal into a sum

* Corresponding author at: Computer Engineering Department, Izmir Institute of Technology, Urla, 35430 İzmir, Turkey. Tel.: +90 232 750 7878.
E-mail address: tolgaayav@iyte.edu.tr (T. Ayav).

http://dx.doi.org/10.1016/j.asoc.2016.06.042
1568-4946/

of periodic functions. Representing a function as a sum of simpler functions allows for a sort of probabilistic reasoning about the various parameters of the system. In our approach, we use fault tree models as input, which are commonly used for sensitivity/importance analysis [2,13,14,16,18,21]. We apply spectral analysis on these models to identify critical components of the architecture. We evaluate our approach based on a benchmark fault tree model and two additional subject models derived from a software architecture description of a Pick and Place Unit (PPU) of a factory automation system [22]. We show that the measures obtained with our approach are both consistent and complementary with respect to the recognized metrics in the literature. Moreover, to the best of our knowledge, this is the first study that applies spectral analysis methods to the evaluation of fault trees.

The remainder of this paper is organized as follows. In the following two sections, we provide background information on fault tree analysis and spectral analysis. In Section 4, we introduce our approach for sensitivity analysis. In Section 5, we present an evaluation of our approach. In Section 6, we discuss the results and limitations. In Section 7, we summarize the related studies. Finally, in Section 8, we conclude the paper.

2. Fault tree analysis

A fault tree is a graphical model, which defines causal relationships among faults leading to a system failure. An example fault tree model is depicted in Fig. 1. Hereby, the top node (i.e., root) or the *top event* of the tree represents the system failure. The leaf nodes of the tree (labeled as *a*, *b*, and *c* in Fig. 1) are named as *basic events*. In our modeling approach, each basic event represents a failure of an individual component of the software architecture. We can also see an *intermediate event* in Fig. 1. Such events represent undesirable system states that can lead to a system failure. Logical connectors, which interconnect the set of events, infer the propagation and contribution of these events to other events and eventually to the system failure. For example, we can see in Fig. 1 that basic events *b* and *c* are connected with an AND-gate (depicted with symbol), which in turn is connected to the intermediate event. This means that the intermediate event occurs if both *b* and *c* occur. This can be the case, for instance, if these basic events represent the failures of functionally equivalent software components employed for N-version programming [21]. Another basic event, *a* is connected with the intermediate event through an OR-gate (depicted with symbol), which in turn is connected to the top event. This means that the top event occurs if one or both of *a* and the intermediate event occur. This can be the case, for instance, if *a* represents the failure of a (critical) component, of which failure directly leads to the system failure regardless of the states of other components.

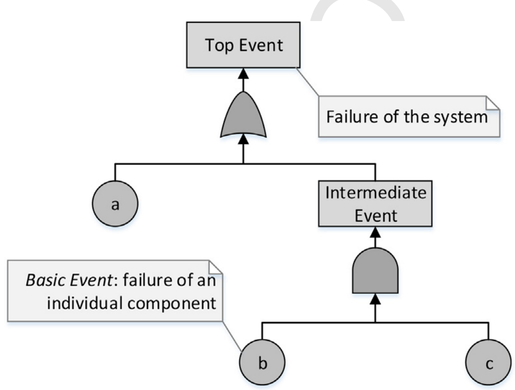

Fig. 1. An example fault tree model.

Occurrence of a set of k events can be represented as a vector of Boolean variables, $\mathbf{x} = [x_0, x_1, \ldots, x_{k-1}]$, of length k. Hereby, $x_i = \mathtt{T}$ and $x_i = \mathtt{F}$ indicate the existence and absence of event i, respectively. Boolean variables and operations are noted and defined as follows:

$$x := F \,|\, T \,|\, x' \,|\, x_1 \Theta x_2, \quad \text{where } \Theta = \{\wedge, \vee\}.$$

An AND-gate represents the intersection of the events attached to the gate. All events must exist for the output event of the gate to occur. For k input events, the equivalent Boolean expression would be

$$f_{\mathrm{AND}}(x) = x_0 \wedge x_1 \wedge x_1 \wedge \cdots \wedge x_{k-1}$$

Let $p_0, \ldots, p_{k-1}$ denote the probabilities of the input events. Under the assumption that these events are independent, the probability of the output event can be defined as

$$p = \prod_{i=0}^{k-1} p_i \tag{1}$$

Similarly, an OR-gate represents the union of the input events. There must exist at least one input event for the output event to occur. The equivalent Boolean expression is

$$f_{\mathrm{OR}}(x) = x_0 \vee x_1 \vee x_1 \vee \cdots \vee x_{k-1}$$

The probability of the output event can be written as

$$p = 1 - \prod_{i=0}^{k-1} (1 - p_i) \tag{2}$$

Let us assume that n different potential failures are identified for a given software architecture. These failures are considered as basic events. Then, a vector of Boolean variables, $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]$ can represent the occurrence of these events. So, $x_i = \mathtt{T}$ and $x_i = \mathtt{F}$ indicate the existence and absence of failure i, respectively. The occurrence of the top event can be represented as a Boolean function of $\mathbf{x}$, $f(\mathbf{x})$, where $f(\mathbf{x}) = \mathtt{T}$ and $f(\mathbf{x}) = \mathtt{F}$ indicate the failure and the correct functioning of the overall system, respectively. For the example fault tree model depicted in Fig. 1, this function can be defined as $f(\mathbf{x}) = a \vee b \wedge c$.

Note that in reliability engineering, failure probabilities depend on time, *i.e.* they are expressed as $p(t)$, $t \in [0, T]$ where T is the mission time and they are assumed to be generated from a failure distribution. Therefore all equations depending on the probabilities are also time dependent. For the sake of clarity, we prefer to use a simpler notation throughout the paper by simply omitting the time dependency. This means that all equations presented in this study are applicable to any time $t \in [0, T]$.

2.1. Coherent and non-coherent systems

Fault tree analysis techniques commonly assume that the analyzed system is a *coherent system*, which is defined as follows:

Definition 1 *(Coherent system).* Given a system with n possible component failures, and its fault tree, where the occurrence of the top event is defined by $f : \mathbb{B}^n \rightarrow \mathbb{B}$, the system is said to be coherent iff:

1. (Relevancy) $\mathrm{Inf}_i > 0 : \quad \forall i \in \{0, 1, 2, \ldots, n-1\}$,
2. (Monotonicity) $f(x) \geq f(y)$ whenever $x \geq y$ pointwise.

The first requirement states that each component must have an influence on whether or not the system works. Second, $f(x)$ is required to be monotone, i.e., a non-decreasing function. In other words, fixing a component cannot make the system worse. Note

Please cite this article in press as: T. Ayav, H. Sözer, Identifying critical architectural components with spectral analysis of fault trees, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.06.042

that a coherent system satisfies that $f(\mathbf{0}) = \mathtt{F}$ and $f(\mathbf{1}) = \mathtt{T}$, meaning respectively if all components are working then the system must be working, and if all components are failed then the system must be failed. Hereby, **0** and **1** represent strings with all bits zero and one, respectively.

Failure probability of the top event can be computed as:

$$P_{TE} = \mathbf{E}[f(x) = T] = \sum_{x=0}^{2^n-1} p(x) \cdot f(x) \tag{3}$$

where $p : \{F, T\}^n \rightarrow \mathbb{R}$ is the probability for a particular value of $\mathbf{x}$ among the 2^n possibilities. $p(x)$ is defined as follows:

$$p(x) = \prod_{i \in x} p_i \cdot \prod_{i \notin x} (1 - p_i) \tag{4}$$

For example, $x = [\mathtt{T}, \mathtt{F}, \mathtt{T}, \mathtt{T}, \mathtt{F}, \mathtt{T}]$ means that the components 0, 2, 3 and 5 are failed at the time of observation, while the other events, 1 and 4 are working. The probability of this case is $p(x) = p_0(1 - p_1)p_2p_3(1 - p_4)p_5$.

It has been observed that the majority of the importance measures are strictly developed for coherent system analysis [23]. On the other hand, some systems can be non-coherent, in which both component failure and recovery contribute to a system failure [24]. This issue was addressed in only a few studies. A non-coherent extension of Birnbaum's importance index [25,24] was previously utilized. A unified framework [23] has been proposed to analyze functions used for obtaining measures of importance and their implications on both coherent and non-coherent systems.

2.2. Dynamic fault trees

Traditional or static fault trees (SFT) capture the effects of a combination of events, i.e., component failures resulting in a system failure. They disregard the temporal order of the occurrence of these events. In some circumstances, the ordering of events do matter. For example, consider a switch that is used for alternating between a component and its spare. The failure of this switch after it activates the spare does not cause a failure; however, its failure before this activation does lead to a system failure. Dynamic fault trees (DFT) were introduced to capture temporal ordering of events and their effects [26].

In SFTs, basic events are represented with Boolean variables. However, Boolean algebra does not have any mechanism to handle the temporal order of occurrence. Hence, DFTs use temporal events as shown in Fig. 2, where $d(a)$ is the unique date of occurrence of event a. Such basic events are supplied as inputs to gates, which model the propagation of the events to a system failure. DFT introduce 3 more gates in addition to the conventional gates used in SFT: the priority AND gate (*PAND*), the spare gate (*SPARE*), and the functional dependency gate (*FDEP*). The commonly used graphical notations for these gates are shown in Fig. 3.

PAND gate is used for capturing failure sequence dependency. All input events must exist for the output event of the PAND gate to occur. In addition, the occurrence of these events must be ordered in time from left to right (i.e., in the order that they are depicted).

SPARE gate is used for representing the management and allocation of spare components. It has one distinguished primary input

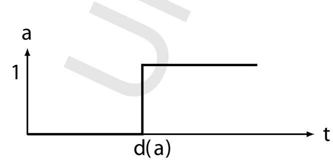

Fig. 2. A non-repairable event.

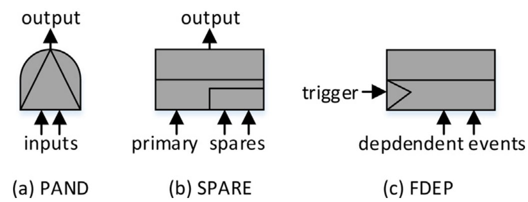

Fig. 3. Dynamic fault tree gates [27].

and one or more spare inputs. If the failure of primary input occurs, it is replaced by one of the spare inputs. The primary and all the spare input events must take place (all the corresponding components must fail) for the output event of the SPARE gate to occur. A spare input can be shared among multiple SPARE gates. In this case, if a spare is already utilized by any of the SPARE gates, it is considered to be unavailable (failed) for the other SPARE gates that share it.

FDEP gate is comprised of a trigger event and a set of dependent events. When the trigger event occurs, it causes the dependent events to occur as well. That is, all the corresponding components will be assumed to be failed. This gate does not have an output, or it has a *dummy* output [28], which is simply ignored. For non-repairable events, it is known that a FDEP gate can be removed by replacing its children by OR gate of the child and the FDEP trigger, and a SPARE gate can be replaced by a k-out-of-N OR gate of SFT [29]. To analyze dynamic fault trees, temporal operators are incorporated [30], as defined below:

$$a \trianglelefteq b = \begin{cases} a & \text{if} \quad d(a) < d(b) \\ a & \text{if} \quad d(a) = d(b) \\ F & \text{if} \quad d(a) > d(b) \end{cases} \tag{5}$$

The symbol $\trianglelefteq$, namely *Inclusive Before* defines the relation among temporal events such that expression $a \trianglelefteq b$ yields a if a non-strictly occurs before b and $\mathtt{F}$ otherwise. The algebraic expression of PAND gate can be written as:

$$a\mathrm{PAND}b = b \wedge (a \trianglelefteq b) \tag{6}$$

For further details on the analysis of fault trees, we refer to the IEC 61025 standard [9] and the literature [31,10]. In the following section, we introduce spectral analysis of boolean functions. Then, we introduce the application of this analysis technique to fault trees.

3. Spectral analysis of boolean functions

Spectral or Fourier analysis is widely used in mathematics and engineering. Fourier decomposes a signal as a sum of periodic functions like $\chi_y(x) = e^{2\pi ixy/n}$. In case of Boolean functions, the most used transform has been defined over Abelian group $\mathbb{Z}_2^n$. Boolean functions are usually defined as $f : \{\mathtt{F}, \mathtt{T}\}^n \rightarrow \{\mathtt{F}, \mathtt{T}\}$. As a requirement of Fourier analysis, instead of 0 and 1, 1 and −1 will be used as False and True values respectively. Hence the function becomes $f : \{1, -1\}^n \rightarrow \{1, -1\}$. The relevant definitions and theorems about Fourier analysis are given below without the proofs. For further explanations, examples and theorems with proofs, we refer to [32–34].

Theorem 1 (Fourier expansion). *Every* $f : \{-1, 1\}^n \rightarrow \mathbb{R}$ *can be expressed with its Fourier expansion,*

$$f(x) = \sum_{\omega \subseteq [n]} \hat{f}(\omega)\chi_\omega(x), \tag{7}$$

Please cite this article in press as: T. Ayav, H. Sözer, Identifying critical architectural components with spectral analysis of fault trees, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.06.042

Table 1
Truth table for $f(x) = a \vee b \wedge c$.

x	a	b	c	$f \in \mathbb{B}$	$f \in \mathbb{R}$
0	F	F	F	F	1
1	F	F	T	F	1
2	F	T	F	F	1
3	F	T	T	T	−1
4	T	F	F	T	−1
5	T	F	T	T	−1
6	T	T	F	T	−1
7	T	T	T	T	−1

where $\hat{f}(\omega)$ is the Fourier coefficient and $\chi_\omega(x) = \prod_{i \in \omega} x_i$ is the parity function. It is also adopted that $\chi_\emptyset \equiv 1$.

Definition 2 *(Inner product).* Let $f, g : \{-1, 1\}^n \to \{-1, 1\}$. The inner product between f and g is defined as

$$\langle f, g\rangle := \sum_{x \in \{-1,1\}^n} \frac{f(x)g(x)}{2^n} = \mathbf{E}_{x \in \{-1,1\}^n}[f(x)g(x)].$$

Note that $\langle f, f\rangle = \|f\|_2^2 = 1$ and more generally $\|f\|_p := \mathbf{E}[|f(x)|^p]^{1/p}$.

Fourier coefficients can be written as

$$\hat{f}(\omega) = \langle f, \chi_\omega\rangle = \mathbf{E}_x[f(x)\chi_\omega(x)]. \tag{8}$$

Note in particular that coefficient $\hat{f}(\emptyset) = \mathbf{E}[f]$ corresponds to the mean $\mathbf{E}[f]$. For example, recall the 3-input Boolean function $f(x) = a \vee b \wedge c$ that was derived in the previous section. Agreeing that a and c are the most and least significant bits respectively, it is trivial to derive the truth vector for f as [FFFTTTTT], i.e., [1 1 1 −1 −1 −1 −1 −1] as shown in Table 1. By using Formula (8), Fourier coefficients can be computed as $\hat{f}(\emptyset) = \frac{1}{2^3}(1+1+1-1-1-1-1-1) = -0.250$, $\hat{f}(1) = \frac{1}{2^3}(1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 - 1 \cdot 1 - 1 \cdot -1 - 1 \cdot -1 - 1 \cdot -1 - 1 \cdot -1) = 0.750$ and so on. The eight coefficients are used to constitute the Fourier expansion of f as follows:

$$\begin{aligned} f &= -0.25 + 0.75a + 0.25b + 0.25ab + 0.25c \\ &\quad + 0.25ac - 0.25bc - 0.25abc. \end{aligned}$$

The derivative or difference calculus for Boolean functions has been benefited in testing digital circuits and also software over the past two decades. It can also be used to describe the notion of influence.

Definition 3 *(Derivative).* The derivative of f with respect to its input x_i is defined as,

$$\frac{\partial f}{\partial x_i} = \frac{f(x_i := -1) - f(x_i := 1)}{(-1) - 1} \tag{9}$$

$$= \sum_{\omega \ni i} \hat{f}(\omega)\chi_{\omega \setminus i}(x). \tag{10}$$

For example, $\frac{\partial f}{\partial a} = \frac{-1-(0.5+0.5b+0.5c-0.5bc)}{-2} = 0.75 + 0.25b + 0.25c - 0.25bc$. It can be noticed that this derivative would produce 0 if b and c are true ($b = c = -1$) and 1 otherwise. f is monotonic, i.e., non-decreasing since changing one bit from false to true would never cause the output to switch from true to false. Monotonicity requirement can also be expressed by $\frac{\partial f}{\partial x_i} \geq 0$, $\forall i$. In theory, all Boolean functions excluding the negation operation are monotonic.

Eq. (9) requires that the derivative of a Boolean function can be in $\{-1, 0, 1\}$. If $\frac{\partial f}{\partial x_i}(x) = \pm 1$, then x_i is said to be pivotal for f at x. If it is zero, then x_i has no influence on f at x. Hence, the influence of input x_i on f is the expected value of being pivotal over x. The definition of influence is as follows:

Definition 4 *(Influence).* The influence of input x_i on f is defined as,

$$\mathrm{Inf}_i(f) := Pr[f(x) \neq f(x^{\oplus i})] = \mathbf{E}_x\left[\frac{\partial f}{\partial x_i}(x)^2\right] = \sum_{\omega \ni i} \hat{f}(\omega)^2. \tag{11}$$

where $x^{\oplus i}$ is the string x with its i-th bit flipped.

For $f(x) = a \vee b \wedge c$, the influence values are found as $\mathrm{Inf}_a(f) = 0.75$ and $\mathrm{Inf}_b(f) = \mathrm{Inf}_c(f) = 0.25$. We can comment on the influence values such that a is the most important component, whereas b and c have identical importance and they are less important with respect to a.

Fault trees are mostly designed to be coherent. The structural functions in this category are therefore monotone. In this case, $\frac{\partial f}{\partial x_i} \geq 0$ and the influence becomes [33]

$$\mathrm{Inf}_i(f) = \mathbf{E}_x\left[\frac{\partial f}{\partial x_i}\right] = \widehat{\frac{\partial f}{\partial x_i}}(\emptyset) = \hat{f}(\{i\}). \tag{12}$$

Eq. (12) implies that the influence of a variable simply equals to one Fourier coefficient. This makes the computation of influences feasible particularly for large functions. For the previous example, $\mathrm{Inf}_a(f) = \hat{f}(1)$, $\mathrm{Inf}_b(f) = \hat{f}(2)$, and $\mathrm{Inf}_c(f) = \hat{f}(4)$.

Another concept is the energy spectrum that may provide usefull information about the noise sensitivity of a function.

Definition 5 *(Energy spectrum).* For any real-valued function $f : \mathbb{B}^n \to \mathbb{R}$, the energy spectrum $\mathbf{E}_f$ is defined by

$$E_f(k) := \sum_{|\omega|=k} \hat{f}(\omega)^2 \qquad \forall k : 1 \leq k \leq n \tag{13}$$

where $|\omega|$ depicts the number of 1 bits in ω.

$E_f(\emptyset)$ is known as DC component, which is the zero-frequency component. If most of the Fourier mass is localized on high frequencies, then the function is sensitive to small perturbations, i.e., component failures as shown in a sample spectrum given in Fig. 4. If a fault tree is found to be noise sensitive, that means, very small number of component faults can significantly impact/degrade the overall system reliability. In that case, one may consider restructuring the architecture to resolve dependencies or dedicate additional effort for adding redundancy and design diversity to improve the reliability.

3.1. Approximating Fourier coefficients

The time and resource usage complexity of the transformation are given as $\mathcal{O}(n2^n)$ and $\mathcal{O}(2^n)$ respectively [35]. Therefore, transformation becomes harder as n gets bigger. In this case, Fourier coefficients can be approximated. Recall that the Fourier coefficient,

$$\hat{f}(\omega) = \mathbb{E}[f \cdot \chi_\omega]$$

is an expectation under uniform distribution. $\chi_\omega : \{0, 1\}^n \to \pm 1$ is a parity function defined as,

$$\chi_\omega(x) = (-1)^{\omega \cdot x},$$

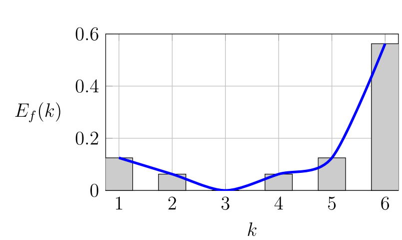

Fig. 4. Energy spectrum of $f(x) = x_0 + x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$.

where $\omega \cdot x = \sum_{i=1}^{n} \omega_i x_i = \sum_{i \in \omega} x_i$. We can approximate the Fourier coefficients from uniformly drawn examples $(x_1, f(x_1)), \ldots, (x_m, f(x_m))$. Expected value is the following empirical average,

$$\frac{1}{m}\sum_{j=1}^{m} f(x_j)\chi_\omega(x_j)$$

and this value converges to the exact value of $\hat{f}(\omega)$ as m grows. Moreover, Chernoff bound tells us how quickly this convergence happens [36].

4. Spectral evaluation of fault trees

In this section, we illustrate how to incorporate spectral analysis into the evaluation of fault trees and propose a new metric for sensitivity analysis. First, we redefine the conventional metrics using Boolean derivative calculus. These metrics are shown to have some drawbacks in particular cases. To overcome these drawbacks, we propose a new metric based entirely on the spectral coefficients. This metric is more informative and it is demonstrated to harmonize the former metrics while eliminating their drawbacks.

Sensitivity metrics can be classified into two categories: structural and probabilistic. The metrics of the former category rely on the location of the component in the failure logical function and the latter category takes into account the failure probability of the associated component. We take into account four common metrics: Birnbaum's importance index, Birnbaum's structural importance index, criticality index and Fussell-Vesely index. Birnbaum's importance index (IB) is widely used and it defines the partial derivative of the system failure probability with respect to the failure probability of its components.

Let $f(x)$ be the Boolean function of a fault tree. Shannon's expansion states that [37]:

$$f(x) = x_i \wedge f_i^T(x) \vee x_{i'} \wedge f_i^F(x) \tag{14}$$

where

$$f_i^T(x) = f(x_1, x_2, \ldots, x_{i-1}, T, x_{i+1}, \ldots, x_n),$$

$$f_i^F(x) = f(x_1, x_2, \ldots, x_{i-1}, F, x_{i+1}, \ldots, x_n).$$

The component is said to be critical if $x_i = \mathtt{T} \rightarrow f(x) = \mathtt{T}$ and $x_i = \mathtt{F} \rightarrow f(x) = \mathtt{F}$. Therefore, the criticality of component x_i can be expressed with the following requirement:

$$f_i^T(x) \wedge f_i^F(x)_{\prime}$$

The probability that this condition holds is

$$Pr[f_i^T(x) \wedge f_i^F(x)_{\prime}] = Pr[f_i^T(x) = T] - Pr[f_i^T(x) \wedge f_i^F(x)_{\prime} = T]^2$$

The Birnbaum index is traditionally given as:

$$IB_i = Pr[f_i^T(x) = T] - Pr[f_i^F(x) = F] \tag{15}$$

We can redefine this metric by exploiting the influence definition of spectral analysis as follows:

Definition 6 *(Birnbaum's importance index (IB)).* Let f be the structure function of a fault tree. Importance index of component i is defined as,

$$IB_i(f) = Pr[f(x) \neq f(x^{\oplus i})] = \mathbf{E}_x\left[\frac{\partial f}{\partial x_i}(x)\right] = \sum_{x=0}^{2^n-1} p(x) \cdot \frac{\partial f}{\partial x_i}(x), \tag{16}$$

where x is a string such that $x \subset [n]$ and $p : \{F, T\}^n \rightarrow \mathbb{R}$ is given as

$$p(x) = \prod_{i \in x} p_i \cdot \prod_{i \notin x}(1 - p_i). \tag{17}$$

In fact, $p(x)$ is the probability of event x. For example, the probability that components b and c are failed at the time of observation, i.e., $x = [\mathtt{F}, \mathtt{T}, \mathtt{T}]$, can be calculated as $p(x) = (1 - p_a)p_b p_c$. Note that the sum of probabilities of all possible event combinations must be 1, i.e., $\sum_{x=0}^{2^n-1} p(x) = 1$. As an example, let $f(x) = a \vee b \wedge c$ represent a fault tree, assuming that the failure probabilities of component a, b and c are given as $p_a = p_b = p_c = 0.1$. We know that $\frac{\partial f}{\partial a} = 0.75 + 0.25b + 0.25c - 0.25bc$ is 1 if $(b, c) \in \{(\mathtt{F}, \mathtt{F}), (\mathtt{F}, \mathtt{T}), (\mathtt{T}, \mathtt{F})\}$. Therefore, $\mathtt{IB}_a(f)$ can be calculated as

$$(0.9 \cdot 0.9) + (0.9 \cdot 0.1) + (0.1 \cdot 0.9) = 0.99.$$

Similarly, we can compute that $\mathtt{IB}_b(f) = \mathtt{IB}_c(f) = 0.09$.

Birnbaum's structural importance index slightly differs from IB relying entirely on the structure of the function. One can realize that it is the equivalent of the influence value defined with Eq. (11). On the other hand, it can also be computed by restricting the definition of IB given with Eq. (16) such that $\Pr[f = \mathtt{F}] = 0.5$, i.e., the probability space consists of all binary n-strings with uniform distribution. In this case, failure probabilities of components are assumed to be identical such that $p_j = 0.5\ \forall j \in \{1, 2, \ldots, n\}$. Hence, $p(x) = 0.5^n$ and the formula in Eq. (16) becomes

$$\mathrm{Inf}_i(f) := \frac{1}{2^n}\sum_{x=0}^{2^n-1} \frac{\partial f}{\partial x_i}(x), \tag{18}$$

which is the expected value given in Eq. (11). As the name implies, this metric depends solely on the structure of the function. It does not take into account the failure probabilities of the components.

Criticality index (IC) [38] represents the probability that the event x_i is critical and its occurrence leads to system failure. It is formulized as

Definition 7 *(Criticality index (IC)).* Let f be the structure function of a fault tree. Criticality index of component i is defined as,

$$IC_i(f) := IB_i(f)\,\frac{p_i}{p_{\mathrm{TE}}}. \tag{19}$$

Note that p_{TE} can be computed as 0.109 using Formula (3). Then, $\mathtt{IC}_a$ is found as follows:

$$IC_a = 0.99 \cdot \frac{0.1}{0.109} = 0.9083.$$

Similarly, we can compute that $\mathtt{IC}_b = \mathtt{IC}_c = 0.0826$.

There exist other metrics proposed in the literature, such as Risk Reduction Worth (RRW), Risk Achievement Worth (RAW) and Fussell-Vesely (FV) [39][40]. RRW measures the change of the system failure probability when a component is perfectly working and is given by:

$$RRW_i(f) = \frac{1}{1 - IC_i(f)} \tag{20}$$

On the contrary, RAW is the measure of the change of system failure probability when a component is supposed to be failed or removed and it can be computed in terms of RRW:

$$RAW_i(f) = \frac{1}{p_i}\left(1 - \frac{1 - p_i}{1 - RRW_i(f)}\right) \tag{21}$$

Since RRW and RAW strictly depend on IC, this study shall not dwell upon them any further. Moreover, they are less expressive than IB and IC in terms of component sensitivity [39]. Nevertheless, FV is rather common in chemical industry and we use it for comparison purposes. FV, referred to as "fractional contribution" is a measure of the contribution of a component to the system failure without being critical. The variable x_i contributes to the system failure when a minimal cut set containing x_i occurs. Therefore, it can be expressed as:

Please cite this article in press as: T. Ayav, H. Sözer, Identifying critical architectural components with spectral analysis of fault trees, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.06.042

Definition 8 *(Fussell-Vesely index* (FV)*).* Let f be the structure function of a fault tree. Fussell-Vesely index of component i is defined as,

$$FV_i(f) = \frac{Pr[\bigcup_s C_s = T]}{P_{TE}} \approxeq \frac{P_{TE} - Pr[f_i^F(x) = T]}{P_{TE}} \approxeq 1 - \frac{1}{P_{TE}} \sum_{x=0}^{2^{n-1}} p(x)(1 - 2f_i^F(x)). \tag{22}$$

For example, $\mathrm{FV}_a(f)$ can be calculated as

$$\frac{0.109 - (0.9 \cdot 0.1 \cdot 0.1)(1 - 2(-1))}{0.109} = 0.9174.$$

Similarly, we can compute that $\mathrm{FV}_b(f) = \mathrm{FV}_c(f) = 0.1743$. The aforementioned four metrics are quite common but they may expose some difficulties or misinterpretation in particular cases. First, even though Inf_i provides useful information about the relation between the component failures and the system failure, it solely seems insufficient to identify critical components since it disregards the failure probabilities of the components. Therefore, IB and IC are often used together for reliability evaluation. For any given component, if both values are low then the associated component is not critical. If both are high, the component can be considered most critical. The case of high IB and low IC, however, indicates that the component is structurally not important or its impact is dominated by other components with high failure probabilities. In this case, one might consider a structural improvement. On the contrary, the case of low IB and high IC indicates a lack of structural flaw, but high failure probability of the associated component. Hence, the component reliability should be improved in this case.

As stated before, IC is expected to involve the failure probability of the associated component, whereas IB is independent of it due to the derivative with respect to that component. However, IC also falls short to reflect failure probability to the evaluation, as will be shown in Section 5. The second drawback for both IB and IC is that they are limited to the analysis of a certain class of systems: *Coherent Systems*. They may induce misleading results for non-coherent systems. Although fault trees are traditionally designed to be coherent, non-coherent fault trees have also been shown useful [41]. Fussell-Vesely is also unable to discriminate the aforementioned two cases: (i) low IB and high IC: The component needs to be improved and (ii) low IC and high IB: The structure function needs to be improved.

In order to overcome these drawbacks, we propose a new importance metric based on Fourier analysis, in which both failure probabilities and IB are taken into account. We call this metric *spectral sensitivity*, which is defined as follows:

Definition 9 *(Spectral Sensitivity).* The spectral sensitivity of input x_i on f is defined as,

$$S_i(f) := \sum_{\omega \ni i} \hat{f}(\omega)^2 p(\omega). \tag{23}$$

where $\hat{f}(\omega)$ depicts spectral of Fourier coefficients and

$$p(\omega) = \prod_{i \in \omega} p_i \cdot \prod_{i \notin \omega} (1 - p_i). \tag{24}$$

Below, we show that S_i is the superposition of Inf_i, IB_i, IC_i and p_i.

Lemma 1. $\sum_{\omega \ni i} p(\omega) = p_i$.

Proof. *We can write* $\sum_{\omega} p(\omega) = p_i \sum_{\omega \ni i} p(\omega \setminus \{i\}) + (1 - p_i) \sum_{\omega \notin i} p(\omega \setminus \{i\}) = 1$. *Note also that* $p_i \sum_{\omega \ni i} p(\omega \setminus \{i\}) = \sum_{\omega \notin i} p(\omega \setminus \{i\}) = 1$. *Hence we can get* $\sum_{\omega \ni i} p(\omega) = p_i \sum_{\omega \ni i} p(\omega \setminus \{i\}) = p_i$.□

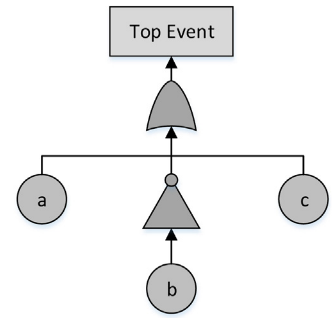

Fig. 5. Example non-coherent fault tree model ($p_a = p_b = p_c = 0.1$).

We know that ω defines a probability distribution over 2^n. Thus $\sum_{\omega} p(\omega) = 1$. By Lemma (1), $\sum_{\omega \ni i} p(\omega) = p_i$. We also have the following facts: By Parseval theorem [32], $\sum_{\omega} \hat{f}(\omega)^2 = 1$ and also by Eq. (11), $\sum_{\omega \ni i} \hat{f}(\omega)^2 = \mathrm{Inf}_i$. One can realize that S_i depends on Inf_i. IB, however, is the weighted average of the failure probability with respect to p_i, so it is a derivative and independent of p_i. Naturally, IB_i depends on Inf_i and by Eq. (19), IC_i depends on IB_i. Note that IC_i may not depend on p_i. However, consider a specific case that component i's order of minimum cut set is 1, i.e., $p_{\mathrm{TE}} = p_i \cdot p_{\mathrm{REST}}$. In this case, p_i disappears in Eq. (19). Let us define a concept of weighted influence such as:

$$\mathrm{WInf}_i = \frac{\sum_{\omega \ni i} p(\omega) \hat{f}(\omega)^2}{\sum_{\omega \ni i} p(\omega)}$$

The denominator equals to p_i. Therefore, this value is expected to be similar to Birnbaum's index IB_i. In order to incorporate p_i into the metric, we can multiply it by p_i, hence one can notice that $\mathrm{S}_i = p_i \mathrm{WInf}_i$, which shows that S_i contains p_i as well.

4.1. Application to non-coherent Systems

A simple fault tree shown in Fig. 5 is used to demonstrate the comparisons of the traditional and spectral measures for non-coherent systems. Let $f = a \vee b' \vee c$ be the Boolean representation of the tree. Note that the system does not satisfy the monotonicity requirement of Definition (1). Since all components have the same order of minimal cut set, Inf_a, Inf_b and Inf_c must be identical. They can be computed as 0.25 expectedly by Eq. (11). Assuming $p_a = p_b = p_c = 0.1$, Birnbaum's indexes can be found as $\mathrm{IB}_a = 0.09$, $\mathrm{IB}_b = -0.81$ and $\mathrm{IB}_c = 0.09$ by Eq. (16). A negative value comes from the partial derivative with respect to b. The probability of b' is $(1 - p_b)$. The derivation of $(1 - p_b)$ with respect to b results in a negative IB value. On the other hand, $|\mathrm{IB}_b|$ is much higher than $|\mathrm{IB}_a|$ and $|\mathrm{IB}_c|$. This indicates that component b is much more critical than a and c, which is misleading. All components are expected to have identical criticality with equal probabilities and influences. Both IB and S measures for this example are shown in Table 2, which confirm this expectation.

4.2. Application to dynamic fault trees

Dynamic fault trees (DFT) involve temporal sequences of failures. Spectral evaluation of Boolean functions does not analyze

Table 2
Importance values of $f = a \vee b' \vee c$.

	a	b	c
Inf_i	0.25	0.25	0.25
p	0.1	0.1	0.1
IB	9e−2	−81e−2	9e−2
S	6.25e−3	6.25e−3	6.25e−3

Please cite this article in press as: T. Ayav, H. Sözer, Identifying critical architectural components with spectral analysis of fault trees, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.06.042

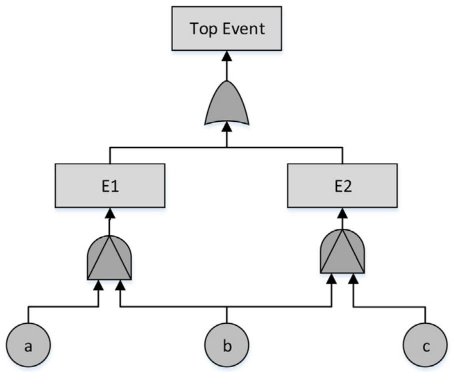

Fig. 6. An example dynamic fault tree model ($p_a = 0.05$, $p_b = 0.06$, $p_c = 0.01$).

the timing behavior in the input combinations, therefore spectral sensitivity measure cannot be applied directly to DFTs. In fact, importance analysis in DFT is still an open research area, and to the best of our knowledge there is no such work. Nevertheless, by altering the definition of Boolean derivative, Birnbaum importance index can still be applied to DFT, yet the nature of this index limits the calculation to a specific mission time. We take into account the analysis for Priority-AND failure logic, since its output depends on the sequence of the inputs. For non-repairable events, it is known that a FDEP gate can be removed by replacing its children by OR gate of the child and the FDEP trigger, and a SPARE gate can be replaced by a k-out-of-N OR gate of static fault trees [29].

Definition 10 *(Time-aware derivative).* Time-aware derivative of f with respect to its input x_i is defined as,

$$\frac{\partial f}{\partial x_i} := \frac{f_i^T(x)^+ - f_i^F(x)^-}{-2} \tag{25}$$

The semantic of + and − notations is such that x_i is assigned true after a false value. The intuition behind this definition is that the output is checked against the switch of input x_i from F to T, as expectedly from a non-repairable component. For $f(x) = a$ AND b, it is easy to check that

$$\frac{\partial f}{\partial a} = \frac{(-1)\cdot b - 1\cdot b}{-2} = b, \quad \frac{\partial f}{\partial b} = \frac{(-1)\cdot a - 1\cdot a}{-2} = a$$

On the other hand, for $f(x) = a$ PAND b, using the notation given in [21], f can be written as

$$f = b \wedge (a \trianglelefteq b)$$

The derivatives with respect to a and b can be found as

$$\frac{\partial f}{\partial a} = \frac{b \wedge (T^+ \trianglelefteq b) - b \wedge (F^- \trianglelefteq b)}{-2} = \frac{b\cdot 1 - b\cdot 1}{-2} = 0$$

Note that $F^- \trianglelefteq b = F$. Also $T^+ \trianglelefteq b = F$, since input a is assigned T later than input b is assigned any value. In this case, PAND produces F regardless of b.

$$\frac{\partial f}{\partial b} = \frac{T^+ \wedge (a \trianglelefteq T^+) - F^- \wedge (a \trianglelefteq F^-)}{-2} = \frac{(-1)\cdot a - 1\cdot a}{-2} = a$$

The example DFT given in Fig. 6 can be expressed as follows:

$$u = (c \wedge (b \trianglelefteq c)) \vee (b \wedge (a \trianglelefteq b)) \tag{26}$$

The derivatives can be computed as,

Table 3
Importance values for $u = c \wedge (b \trianglelefteq c) \vee b \wedge (a \trianglelefteq b)$.

	a	b	c
Inf_i	0	0.5	0.25
p	0.05	0.06	0.01
IB	0	5e−2	5.7e−2

$$\frac{\partial u}{\partial a} = \frac{(c \wedge (b \trianglelefteq c)) \vee b \wedge (T^+ \trianglelefteq b) - (c \wedge (b \trianglelefteq c)) \vee b \wedge (F^- \trianglelefteq b)}{-2} = 0 \tag{27}$$

$$\begin{aligned}\frac{\partial u}{\partial b} &= \frac{(c \wedge (T^+ \trianglelefteq c)) \vee T^+ \wedge (a \trianglelefteq T)]\prime - (c \wedge (F^- \trianglelefteq c)) \vee F^- \wedge (a \trianglelefteq F^-)}{-2}\\ &= 0.5 - 0.5a\end{aligned} \tag{28}$$

$$\frac{\partial u}{\partial c} = 0.25 + 0.25a - 0.25b - 0.25ab \tag{29}$$

The combinations of (a, b, c) that make $\frac{\partial u}{\partial b} = 1$ are (T, F, F), (T, T, F), (T, F, T) and (T, T, T). Therefore, IB_c can be computed as:

$$\begin{aligned}&0.5(1-0.06)(1-0.25) + 0.5 \quad 0.06(1-0.25) + 0.5(1-0.06)0.25\\ &+ 0.5 \quad 0.06 \quad 0.25 = 0.5.\end{aligned}$$

Table 3 shows the complete results for this tree. Note that at a specific mission time, IB_a is zero. Consider these two cases: (i) a fails (switches to T) when b is working and (ii) a fails after b has failed. In both cases, the output will remain zero. Therefore, failure of a itself does not have any effect on the output according to the definition of Birnbaum. Component c seems to be most critical component among these three.

5. Evaluation

This section will describe the evaluation of the approach and results. First, we evaluate our approach with a benchmark fault tree model given in [42]. Then, we apply our approach on two larger fault tree models. These models are derived from a software architecture description of a Pick and Place Unit (PPU) of a factory automation system [22].

Fig. 7 illustrates the model that is used as a benchmark [42]. The structure function of the tree can be constructed step by step as follows:

$$E11 = x_0 \cdot x_1 \tag{30}$$

$$E22 = x_4 + x_5 \tag{31}$$

$$E1 = E11 + x_2 = x_0 x_1 + x_2 \tag{32}$$

$$E2 = x_3 \cdot E22 = x_3(x_4 + x_5) \tag{33}$$

$$f(x) = E1 \cdot E2 = (x_0 x_1 + x_2) x_3 (x_4 + x_5). \tag{34}$$

We can evaluate the tree structurally, using the influences (Birnbaum's structural importance) of the basic events and the energy spectrum of the function. The influence values Inf_i are given in Table 4. The influences are independent of the probabilities, hence they remain the same in all the experiments. All influence values are non-zero and $f(x)$ is monotonic, therefore the system is coherent. According to Inf_i, x_3 is the most important component, followed by x_2. We can also see the energy spectrum of the tree in Fig. 8. Since the Fourier mass is localized at the low frequency (left) side, one can conclude that the tree is not noise sensitive. This means that the system failure depends on the failure of many components.

Please cite this article in press as: T. Ayav, H. Sözer, Identifying critical architectural components with spectral analysis of fault trees, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.06.042

Table 4
Evaluation results.

	x_0	x_1	x_2	x_3	x_4	x_5
Inf_i	0.094	0.094	0.281	**0.469**	0.156	0.156
Experiment 1						
p	0.05	0.06	0.01	0.02	0.04	0.03
`IB`	8.17344e−5	6.8112e−5	**1.371872e−3**	8.92336e−4	2.51618e−4	2.49024e−4
`IC`	2.2899e−1	2.2899e−1	7.68697e−1	**1.0e+0**	5.639535e−1	4.186047e−1
`FV`	2.675405e−01	2.752506e−01	7.71010e−01	**1.0e+0**	5.813953e−01	4.360465e−01
`S`	4.125781e−4	4.950937e−4	6.720072e−4	**3.678401e−3**	8.706687e−4	6.530016e−4
Experiment 2						
p	0.05	0.06	0.01	0.02	0.04	**0.2**
`IB`	2.75616e−4	2.2968e−4	**4.62608e−3**	3.00904e−3	2.0752e−4	2.49024e−4
`IC`	2.2899e−1	2.2899e−1	7.68697e−1	**1.0e+0**	1.37931e−1	8.275862e−1
`FV`	2.675405e−01	2.752506e−01	7.710100e−01	**1.0e+0**	1.724138e−01	8.620690e−01
`S`	3.488281e−4	4.185938e−4	5.681712e−4	3.110029e−3	8.706687e−4	**4.353344e−3**
Experiment 3						
p	0.05	0.06	0.01	**0.001**	0.04	0.03
`IB`	4.08672e−6	3.4056e−6	6.85936e−5	**8.92336e−4**	1.25809e−5	1.24512e−5
`IC`	2.2899e−1	2.2899e−1	7.68697e−1	**1.0e+0**	5.639535e−1	4.186047e−1
`FV`	2.675405e−01	2.752506e−01	7.710100e−01	**1.0e+0**	5.813953e−01	4.360465e−01
`S`	4.125781e−4	4.950937e−4	6.720072e−4	1.839201e−4	**8.706687e−4**	6.530016e−4

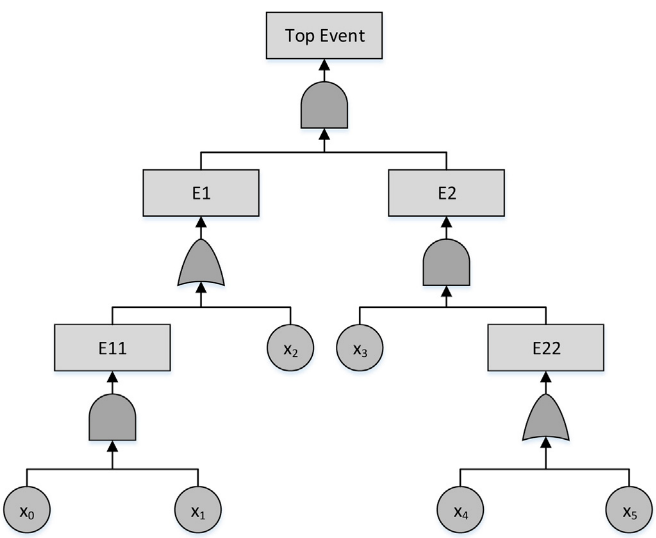

Fig. 7. Fault tree model used as a benchmark [42].

Taking on the probability values of Experiment 1, we can calculate the failure probability of the top event by Eq. (3):

$$P_{\mathrm{TE}} = 1,784672 \cdot 10^{-5}.$$

We applied three different parameter settings on the given fault tree model to compare different sensitivity metrics, including the one we proposed. All the results are presented in Table 4.

The first setting is the same with [42]. In this experiment, according to `IB`, x_2 is the most important component, followed by x_3. In terms of criticality index, component 3 is most critical with $\mathtt{IC}_3 = 1.0$, followed by x_2. Moreover this value does not respond to the probability changes as will be shown next. We can calculate the failure probability of the top event as: $P_{\mathrm{TE}} = (p_0p_1 + p_2 - p_0p_1p_2)p_3(p_4 + p_5 - p_4p_5)$ by Eqs. (1) and (2). Thus,

$$IC_3 = \frac{\partial P_{\mathrm{TE}}}{\partial p_3}\frac{p_3}{P_{\mathrm{TE}}} = ((p_0p_1 + p_2 - p_0p_1p_2)(p_4 + p_5 - p_4p_5))\frac{p_3}{P_{\mathrm{TE}}} = 1.$$

Hence, $\mathtt{IC}_3$ is independent of the probabilities. When the failure probability of component 3 is extremely low, this value becomes misleading as demonstrated in the results of the third experiment listed in Table 4. The spectral sensitivity indicates x_3 as the most important component yet the second is x_4 unlike the results of `IB` and `IC`. This is because the probability of x_4 is four times the one of x_2.

In the second experiment, we increase the failure probability of x_5 6.67 times to see the effect of `S`. According to `IB` and `IC`, the most important components remain the same, where `S` now points out x_5.

In the third experiment, we decrease p_3 20 times. Although p_3 is quite smaller than the other probabilities, this time `IB` and `IC` indicate x_3 as the most important, which is again misleading. On the other hand, `S` orders the first three components as x_4, x_2 and x_5. The results show that `S` takes into account `IB`, `IC` and the failure probabilities of components as well.

We can also see in Table 4 that `IC` and `FV` yield to same results for all the three experiments. In the following, we evaluate our approach with two larger fault tree models. We apply three different parameter settings on these and perform comparisons among `IB`, `IC` and `S`.

The analyzed fault tree models are derived from two different versions of a software architecture description of a Pick and Place Unit (PPU) of a factory automation system [22] that was evolved over time. These models are enumerated as *FT1-SC14* and *SC0-10*. They are depicted in Figs. 9 and 11, respectively. We used uniformly distributed random numbers as the probability values of basic events.

The sensitivity results and the energy spectrum of FT1-SC14 are presented in Table 5 and Fig. 10 respectively. In this system, all importance metrics agree on the same component, x_3 as the most critical one. The reason is that the structural importance order of basic events is as follows:

$$x_0 = x_1 = \ldots = x_8 = x_{14} = x_{15} > x_9 = x_{10} > x_{14} = x_{15},$$

therefore the component with the highest failure probability is expected to be most critical. The energy spectrum exhibits a more

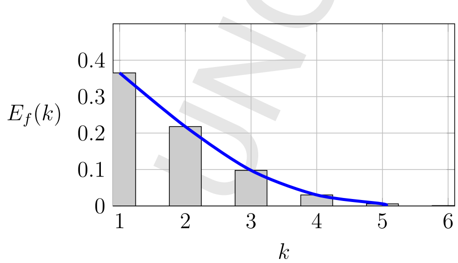

Fig. 8. Energy spectrum of $f(x) = (x_0x_1 + x_2)x_3(x_4 + x_5)$.

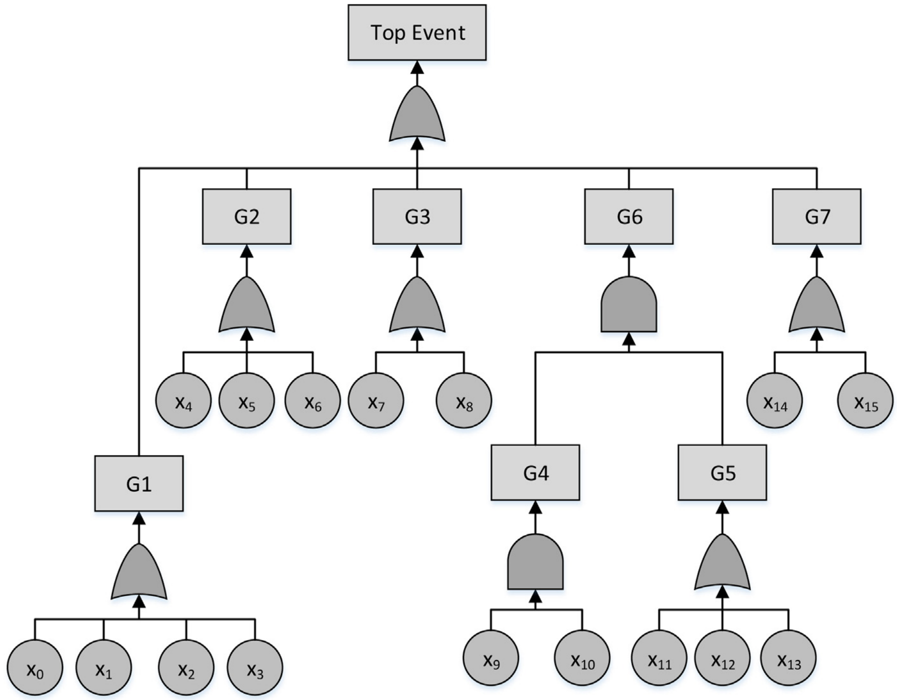

Fig. 9. Fault tree model derived from a software architecture (FT1-SC14) [22].

pessimistic view than the previous benchmark. The Fourier mass is localized in the middle, which means that the top event depends on very few basic events. This can be verified from Fig. 9. Most of the basic events are connected to the top event through OR gates and a failure of one of them is sufficient to cause the top event.

The results of SC0-10 are presented in Table 6. Here, `IB` and `IC` point out x_{14} as the most critical component whereas x_1 has the highest spectral sensitivity. The energy spectrum of SC0-10, shown in Fig. 12 has a quite similar characteristic with the previous spectrum, *i.e.* very few component failures may cause a system failure.

6. Discussion

Regarding the validity of our evaluation results, we can consider 4 types of threats [43]: conclusion validity, construct validity, internal validity and external validity. Conclusion and construct validity threats are mitigated by comparing our results with respect to well-established metrics in the literature. To overcome internal validity issues, we obtained our subject models that are previously published and utilized for other experiments. External validity is related to the representativeness of the selected fault tree models, which is necessary for generalizing the results. We used three different models to mitigate this threat.

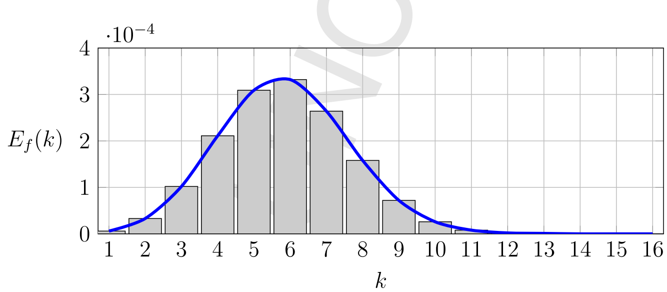

Fig. 10. Energy spectrum of FT1-SC14.

Our approach is subject to limitations when it is employed in a dynamic context such that the fault tree model and/or component reliabilities change in time. If the fault tree model changes due to the evolution/adaptation of the architecture [22], metric evaluation should be repeated. Likewise, if a set of design alternatives has to be evaluated, metric calculations should be performed for each alternative design as a whole to make a comparison among them. Our approach is indifferent with respect to the other sensitivity/criticality metrics from this perspective. There exist reliability analysis approaches [28] that are particularly focusing on facilitating modular analysis and as such the (partial) reuse of calculations in case of changes. We did not consider this issue in our work. Similarly, component reliabilities are assumed to be crisp and constant values. This assumption may not be valid for all type of components especially when runtime adaptations are possible. However, one can repeat calculations for a range of values to perform a what-if analysis. Such metric evaluations can be pre-computed offline, especially if the potential changes can be predicted. These computations can be used at runtime depending on the observed changes.

The time complexity of the calculation of the entire spectra for a Boolean function is denoted with $\mathcal{O}(n2^n)$. Therefore, the complexity of our spectral sensitivity metric given in Eq. (23) is $\mathcal{O}(n^2 2^n)$. Traditional metrics are usually calculated either using Boolean manipulation or through Binary Decision Diagram (BDD) representation of the fault trees [29]. Conversion to a BDD has exponential worst-case and linear best-case complexity. However, BDDs are shown to exhibit better performance than Boolean manipulation since they provide a compact representation of Boolean functions with a high degree of symmetry and fault trees show this symmetry. Once a BDD is obtained, cut sets can be determined by starting at all 1-leaves and traversing upwards to the root. Birnbaum and other aforementioned metrics can be calculated through the cut sets, therefore they have linear complexity after the BDD is generated. In all conditions, spectral techniques appear to be inefficient compared to the other methods. Nevertheless, the approximation method presented in Section 3.1 is

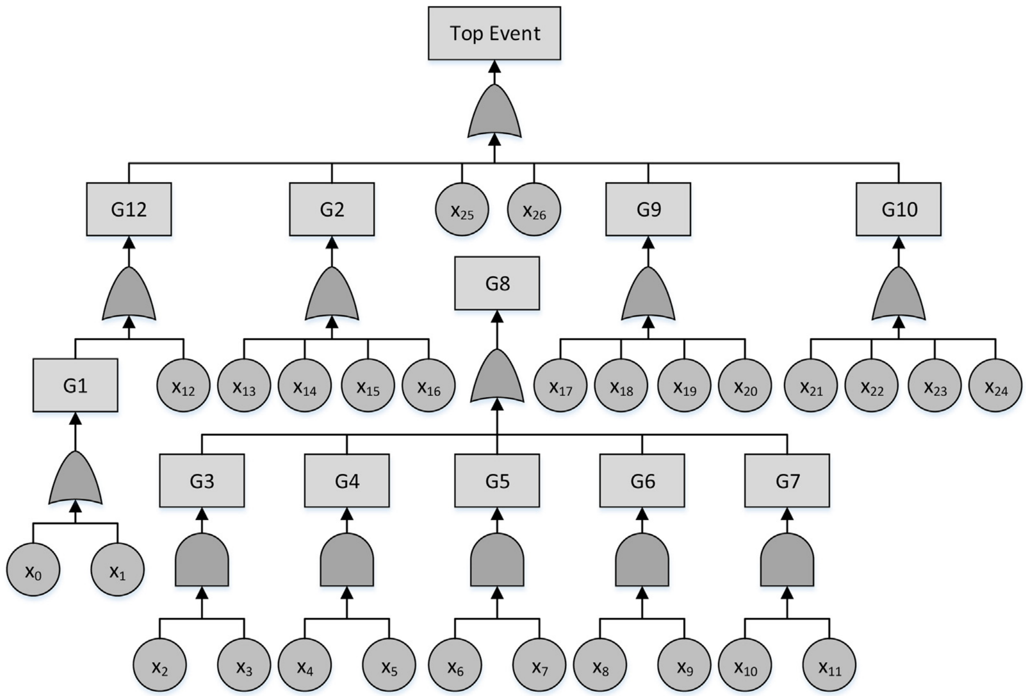

Fig. 11. Fault tree model derived from a software architecture (SC0-10) [22].

Table 5
Evaluation results for FT1-SC14.

	x_0	x_1	x_2	x_3	x_4	x_5
Inf_i	7.629395e−04	7.629395e−04	7.629395e−04	7.629395e−04	7.629395e−04	7.629395e−04
p	0.035457	0.035338	0.061671	0.094864	0.035846	0.032513
IB	6.128847e−01	6.128092e−01	6.300071e−01	**6.531107e−01**	6.131324e−01	6.110200e−01
IC	5.315184e−02	5.296724e−02	9.503183e−02	**1.515409e−01**	5.375765e−02	4.859102e−02
S	1.598650e−08	1.593293e−08	2.780592e−08	**4.277172e−08**	1.616217e−08	1.465934e−08

	x_6	x_7	x_8	x_9	x_{10}	x_{11}
Inf_i	7.629395e−04	7.629395e−04	7.629395e−04	2.136230e−04	2.136230e−04	3.051758e−05
p	0.035781	0.021062	0.049430	0.096217	0.031884	0.039217
IB	6.130906e−01	6.038726e−01	6.218937e−01	2.352263e−03	7.098438e−03	1.652711e−03
IC	5.365563e−02	3.110920e−02	7.518718e−02	5.535748e−04	5.535748e−04	1.585302e−04
S	1.613260e−08	9.496362e−09	2.228647e−08	3.854247e−09	1.277211e−09	3.652374e−11

	x_{12}	x_{13}	x_{14}	x_{15}
Inf_i	3.051758e−05	3.051758e−05	7.629395e−04	7.629395e−04
p	0.003908	0.085452	0.078028	0.030211
IB	1.594126e−03	1.736263e−03	6.411842e−01	6.095694e−01
IC	1.523695e−05	3.628915e−04	1.223699e−01	4.504311e−02
S	3.639449e−12	7.958321e−11	3.518079e−08	1.362131e−08

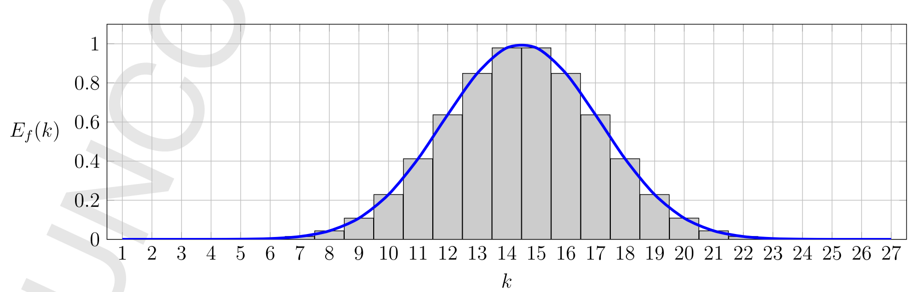

Fig. 12. Energy spectrum of SC0-10.

Please cite this article in press as: T. Ayav, H. Sözer, Identifying critical architectural components with spectral analysis of fault trees, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.06.042

Table 6
Evaluation results for SC0-10.

	x_0	x_1	x_2	x_3	x_4	x_5	x_6
p	0.048211	0.090476	0.009704	0.073332	0.047763	0.046852	0.006192
IB	4.14e-01	4.33e-01	3.98e-01	0.00e+00	1.85e-02	1.89e-02	1.06e-03
IC	3.30e-02	6.47e-02	6.38e-03	0.00e+00	1.46e-03	1.46e-03	1.08e-05
S	1.05e-02	**1.58e-02**	7.54e-04	2.00e-04	2.20e-04	2.14e-04	7.49e-05

	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	
p	0.002682	0.059671	0.092333	0.038408	0.069307	0.002909	0.064377
IB	2.44e−03	3.66e−02	2.37e−02	2.74e−02	1.52e−02	0.00e+00	4.21e−01
IC	1.08e−05	3.61e−03	3.61e−03	1.74e−03	1.74e−03	0.00e+00	4.48e−02
S	1.04e−05	1.71e−04	3.40e−04	1.00e−04	1.95e−04	1.01e−05	1.68e−04

	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
p	0.09597	0.000987	0.019109	0.087771	0.053134	0.090643	0.020308
IB	**4.36e−01**	3.95e−01	4.02e−01	4.32e−01	4.16e−01	4.34e−01	4.02e−01
IC	**6.91e−02**	6.43e−04	1.27e−02	6.26e−02	3.65e−02	6.49e−02	1.35e−02
S	2.82e−04	5.34e−06	6.23e−05	2.19e−04	1.66e−04	5.50e−04	7.16e−05

	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}
p	0.013639	0.024008	0.056297	0.090061	0.064624	0.05769
IB	4.00e−01	4.04e−01	4.18e−01	4.33e−01	4.22e−01	4.18e−01
IC	9.00e−03	1.60e−02	3.88e−02	6.44e−02	4.50e−02	3.98e−02
S	5.62e−05	7.70e−05	1.90e−04	1.88e−04	2.98e−04	2.63e−04

promising since it allows to compute the coefficients in reasonable times. Moreover, approximate values are sufficient to find the order of importance of the components. In the evaluation section, the spectral sensitivities for the two relatively large fault trees, FT1-SC14 and SC0-10, are calculated with the approximate spectral coefficients.

7. Related work

There have been many software architecture analysis techniques introduced [6]. These techniques mainly adopt *scenario-based analysis* approaches. Hereby, the impact of set scenarios is analyzed on a model of the architecture to identify the potential risks and the sensitive points of the architecture. Different analysis methods use different type of scenarios (e.g., usage scenarios [44], change scenarios [45], failure scenarios [18]) depending on the quality attributes (e.g, performance, maintainability, reliability) that they focus on. Some methods such as ATAM [44] utilize multiple types of scenarios for addressing multiple quality attributes at the same time. Previously, failure scenarios have been used for analyzing software reliability at the architecture design level [18]. In that approach, fault tree models have been defined based on these scenarios [14]. Then, sensitivity analysis has been performed on these models based on the measure introduced by Birnbaum [19]. In this work, we assume that the fault tree model of the system is given as input. However, we apply a new technique for sensitivity analysis.

In this work, we applied spectral analysis of Boolean functions for sensitivity analysis. There also exist other sensitivity analysis approaches that are applied based on (dynamic) fault tree models [16,5]. These approximate approaches make use of variations of Markov chain models (DTMC, CTMC) in order to model the software architecture. These models are usually derived based on fault trees [16,46] that are provided as input. As an alternative approach to analytical resolution, there also exist simulation techniques and tools [47] applied on fault tree models. They are used particularly for analyzing DFT models to perform dynamic reliability assessment [48,49]. They mainly employ Monte Carlo simulation with the aim of overcoming the limitations of analytical methods such as state space explosion and lack of modularity in analysis [47]. There also exist studies that benefit from Bayesian Network (BN) that is a powerful method for probabilistic reasoning, particularly taking into account the complex dependencies in components and uncertainty in modeling. To that end both FT and DFT can be converted into BN [50] and Dynamic BN [51] respectively.

Spectral analysis of Boolean functions is a powerful technique. In the literature, Fourier analysis, Walsh or Walsh-Hadamard transformations, Reed-Muller transformation are all used for spectral analysis. These techniques are well-known for more than thirty years. Although they have a wide application area in mathematics, physics and engineering, its application in computer science seems relatively limited. Some fields that have utilized spectral analysis so far include error-correcting code analysis, cryptography, graph theory and quantum computing. Spectral analysis of Boolean functions has attracted a great attention from computer scientists in the last decade [32–34]. This is due to well-developed theorems such as Kahn-Kalai, Arrow's and Peres's theorems, and also its contribution in the development of social choice theory. The influence of Boolean variables and noise sensitivity has also been studied by several papers [32,34,52]. In our study, we benefit particularly from influence, probabilistic influence and energy spectrum of spectral analysis in order to analyze fault trees that represent both coherent and non-coherent systems.

Our spectral analysis technique can be safely used for both coherent and non-coherent systems. IEC 61025 does not distinguish between these two types systems [9]. Indeed, fault tree analysis is usually applied to coherent systems. However, as stated by [25,41], non-coherent systems can also be useful in many cases and the sensitivity analysis techniques proposed for these systems are quite limited. It is also demonstrated that the conventional metrics, IB and IC provide misleading results for the evaluation of non-coherent systems. Therefore, some extensions have been proposed for these metrics [25,24]. There were also other extensions proposed [53] to address complex components (as well as group of components) whose failures are triggered by a combination of basic events. In our approach, component failures are modeled in the form of basic events.

8. Conclusions, limitations and future work

We introduced a new approach for identifying critical components of the software architecture with respect to reliability. Our approach employs a spectral analysis of fault trees that are commonly used models for sensitivity and importance analysis at the architecture design level. The approach is applied on benchmark fault tree models and the results are compared with respect to the recognized metrics in the literature. It was observed that the measures obtained with our approach are consistent with the existing metrics. In addition, we showed that our approach can facilitate the analysis of different types of fault trees, which are considered to be out-of-scope for the current metrics.

Our approach is currently applicable to static fault trees only. Further research is necessary for extending or complementing the approach to make it applicable for dynamic fault trees as well. Likewise, it is assumed that the software architecture is not subject to changes and component reliabilities are defined as crisp and constant values. Another limitation of the approach is time complexity leading to exponential growth in the worst case. We offered an approximation method to be able compute coefficients in reasonable times. In our experiments, approximate values turned out to be sufficient for finding the relative component importance. However, more case studies and controlled experiments are needed to evaluate the effectiveness of our approximation method for different types of subject systems.

Acknowledgements

This study was partially supported by research grant 115E726 from the Scientific and Technological Research Council of Turkey.

References

[1] S. Gokhale, Architecture-based software reliability analysis: overview and limitations, IEEE Trans. Softw. Eng. 4 (1) (2007) 32–40.
[2] A. Immonen, E. Niemela, Survey of reliability and availability prediction methods from the viewpoint of software architecture, Softw. Syst. Model. 7 (1) (2008) 49–65.
[3] P. Clements, et al., Documenting Software Architectures: Views and Beyond, Addison-Wesley, 2002.
[4] M. Shaw, R. DeLine, D. Klein, T. Ross, D. Young, G. Zelesnik, Abstractions for software architecture and tools to support them, IEEE Trans. Softw. Eng. 21 (4) (1995) 314–335.
[5] S. Gokhale, K. Trivedi, Reliability prediction and sensitivity analysis based on software architecture, in: Proceedings of the 13th International Symposium on Software Reliability Engineering, 2002, pp. 64–76.
[6] E.N.L. Dobrica, A survey on software architecture analysis methods, IEEE Trans. Softw. Eng. 28 (7) (2002) 638–654.
[7] W.-L. Wang, D. Pan, M.-H. Chen, Architecture-based software reliability modeling, J. Syst. Softw. 79 (1) (2006) 132–146.
[8] K. Goeva-Popstojanova, K. Trivedi, Architecture-based approaches to software reliability prediction, Comput. Math. Appl. 46 (7) (2003) 1023–1036.
[9] IEC61025:2006, Fault tree analysis (FTA), in: The International Electrotechnical Commission (IEC), Geneva, Switzerland, 2006 https://webstore.iec.ch/publication/4311.
[10] M.A. Boyd, Dynamic fault tree models: techniques for analyses of advanced fault tolerant computer systems (Ph.D. thesis), Duke University, Durham, USA, 1991.
[11] H. Sun, M. Hauptman, R. Lutz, Integrating product-line fault tree analysis into AADL models, in: Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium, 2007, pp. 15–22.
[12] C. Lauer, R. German, J. Pollmer, Fault tree synthesis from uml models for reliability analysis at early design stages, SIGSOFT Softw. Eng. Notes 36 (1) (2011) 1–8.
[13] H. Sozer, B. Tekinerdogan, M. Aksit, Extending failure modes and effects analysis approach for reliability analysis at the software architecture design level, in: R. de Lemos, C. Gacek, A. Romanovsky (Eds.), Architecting Dependable Systems IV, Vol. 4615 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2007, pp. 409–433.
[14] H. Sozer, Architecting fault-tolerant software systems (Ph.D. thesis), University of Twente, Enschede, The Netherlands, 2009.
[15] E. Borgonovo, E. Plischke, Sensitivity analysis: a review of recent advances, Eur. J. Oper. Res. 248 (3) (2016) 869–887.
[16] Y. Ou, B. Dugan, Sensitivity analysis of modular dynamic fault trees, in: Proceedings of the IEEE International Symposium on Computer Performance and Dependability, 2000, pp. 35–43.
[17] W. Kuo, X. Zhu, Importance Measures in Reliability, Risk and Optimization – Principles and Applications, Wiley, Hoboken, NJ, USA, 2012.
[18] B. Tekinerdogan, H. Sozer, M. Aksit, Software architecture reliability analysis using failure scenarios, J. Syst. Softw. 81 (4) (2008) 558–575.
[19] J. Andrews, T. Moss, Reliability and Risk Assessment, Longman Scientific and Technical, Essex, 1993.
[20] E. Henley, H. Kumamoto, Probabilistic Risk Assessment, IEEE Press, 1992.
[21] J. Dugan, Software system analysis using fault trees, in: M.R. Lyu (Ed.), Handbook of Software Reliability Engineering, McGraw-Hill, 1996, pp. 615–659, chapter 15.
[22] S. Getir, M. Tichy, A. van Horn, L. Grunske, Co-evolution of software architecture and fault tree models: an explorative case study on a pick and place factory automation system, in: Proceedings of the 5th International Workshop on Non-functional Properties in Modeling, 2013, pp. 32–39.
[23] E. Borgonovo, The reliability importance of components and prime implicants in coherent and non-coherent systems including total-order interactions, Eur. J. Oper. Res. 204 (3) (2010) 485–495.
[24] S. Beeson, J. Andrews, Importance measures for non-coherent-system analysis, IEEE Trans. Reliab. 52 (3) (2003) 301–310.
[25] S.C. Beeson, Non coherent fault tree analysis (PhD thesis), Loughborough University, 2002 https://dspace.lboro.ac.uk/2134/6927.
[26] J.B. Dugan, S.J. Bavuso, M.A. Boyd, Dynamic fault-tree models for fault-tolerant computer systems, IEEE Trans. Reliab. 41 (3) (1992) 363–377.
[27] S. Junges, D. Guck, J. Katoen, A. Rensink, M. Stoelinga, Fault trees on a diet – automated reduction by graph rewriting, in: Proceedings of the 1st International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, 2013, pp. 3–18.
[28] H. Boudali, P. Crouzen, M. Stoelinga, Dynamic fault tree analysis using input/output interactive Markov chains, in: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007, pp. 708–717.
[29] E. Ruijters, M. Stoelinga, Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools, Comput. Sci. Rev. 15 (2015) 29–62, http://dx.doi.org/10.1016/j.cosrev.2015.03.001.
[30] G. Merle, J. Roussel, J. Lesage, A. Bobbio, Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events, IEEE Trans. Reliab. 59 (1) (2010) 250–261, http://dx.doi.org/10.1109/TR.2009.2035793.
[31] W. Veseley, F. Goldberg, N. Roberts, D. Haasl, Fault tree handbook, Tech. Rep. NUREG-0492, United States Nuclear Regulatory Commission, Washington, DC, 1981.
[32] R. O'Donnell, Some topics in analysis of Boolean functions, in: Proceedings of the fortieth annual ACM symposium on Theory of computing – STOC 08, 2008, p. 569, http://dx.doi.org/10.1145/1374376.1374458.
[33] R. O'Donnell, P. Austrin, Analysis of Boolean Functions: Notes from a series of lectures by Ryan O'Donnell, Workshop on Computational Complexity, 2012 arXiv:1205.0314v1.
[34] R.D. Wolf, A brief introduction to Fourier analysis on the Boolean cube, Theory Comput. Grad. Surv. 1 (015848) (2008) 1–20, http://dx.doi.org/10.4086/toc.gs.2008.001.
[35] P. Porwik, Efficient spectral method of identification of linear Boolean function, Control Cybern. 33 (4) (2004) 83–105.
[36] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge University Press, New York, NY, 2005.
[37] C.E. Shannon, The synthesis of two terminal switching circuits, Bell Syst. Tech. J. 28 (1) (1949) 59–98 http://bstj.bell-labs.com/BSTJ/images/Vol28/bstj28-1-59.pdf.
[38] S. Contini, V. Matuzas, Components' importance measures for initiating and enabling events in fault tree analysis, Tech. Rep. EUR 24373, European Commission, Joint Research Centre, Luxembourg, 2010.
[39] C. Sergio, F. Luciano, M. Vaidas, Concurrent importance and sensitivity analysis applied to multiple fault trees, 2009.
[40] M. Rausand, A. Hyland, Component Importance, John Wiley & Sons, Inc, 2008, pp. 183–206, http://dx.doi.org/10.1002/9780470316900.ch5.
[41] S. Contini, G. Cojazzi, G. Renda, On the use of non-coherent fault trees in safety and security studies, Reliab. Eng. Syst. Saf. 93 (12) (2008) 1886–1895, http://dx.doi.org/10.1016/j.ress.2008.03.018.
[42] P. Lszl, Sensitivity investigation of fault tree analysis with matrix-algebraic method, Theory Appl. Math. Comput. Sci. 1 (1) (2011) 34–44.
[43] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering, Springer-Verlag, Berlin, Heidelberg, 2012.
[44] P. Clements, R. Kazman, M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley, 2002.
[45] P. Bengtsson, N. Lassing, J. Bosch, H. van Vliet, Architecture-level modifiability analysis (ALMA), J. Syst. Softw. 69 (1-2) (2004) 129–147.
[46] Y. Ou, B. Dugan, Approximate sensitivity analysis for acyclic Markov reliability models, IEEE Trans. Reliab. 52 (2) (2003) 220–230.
[47] G. Manno, F. Chiacchio, L. Compagno, D. D'Urso, N. Trapani, Matcarlore: an integrated {FT} and Monte Carlo Simulink tool for the reliability assessment of dynamic fault tree, Expert Syst. Appl. 39 (12) (2012) 10334–10342.
[48] F. Chiacchio, D. D'Urso, G. Manno, L. Compagno, Stochastic hybrid automaton model of a multi-state system with aging: reliability assessment and design consequences, Reliab. Eng. Syst. Saf. 149 (2016) 1–13.
[49] F. Chiacchio, D. D'Urso, L. Compagno, M. Pennisi, F. Pappalardo, G. Manno, Shyfta, a stochastic hybrid fault tree automaton for the modelling and simulation of dynamic reliability problems, Expert Syst. Appl. 47 (2016) 42–57.

[50] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, Improving the analysis of dependable systems by mapping fault trees into Bayesian networks, Reliab. Eng. Syst. Saf. 71 (3) (2001) 249–260.
[51] L. Portinale, A. Bobbio, D. Raiteri, S. Montani, Compiling dynamic fault trees into dynamic Bayesian nets for reliability analysis: the RADYBAN tool, in: Proceedings of the 5th UAI Bayesian Modeling Applications Workshop, 2007.
[52] J. Kahn, G. Kalai, N. Linial, The influence of variables on Boolean functions, in: 29th Annual Symposium on Foundations of Computer Science, 1998, http://dx.doi.org/10.1109/SFCS.1988.21923.
[53] Y. Dutuit, A. Rauzy, On the extension of importance measures to complex components, Reliab. Eng. Syst. Saf. 142 (2015) 161–168.