Journal of Web Engineering, Vol. 15, No.1&2 (2016) 001-028
© Rinton Press

# DERIVING FAULT TRIGGER METRIC FOR WEB BASED SYSTEMS

SANGEETA SABHARWAL RITU SIBAL
*Netaji Subhas Institute of Technology,*
*Department of Computer Science and Information Technology*

CHAYANIKA SHARMA
*University of Delhi, Delhi, India*
*Email: chayanika_29a@yahoo.com*

Received December 15, 2014
Revised June 12, 2015

Important issues regarding web applications are measuring the complexity and reliability of the system and testing every possible sequence of events. Hence, there is a need to identify and analyze the potential failures of the system. In the current research work, the concept of web links and Fault Tree Analysis (FTA) technique has been used to identify the potential failures of a web application. The web application is divided into modules and possible faults in each module are represented by a fault tree. Each fault event in a fault tree is assigned a measure using number of static links or dynamic links calculated using a metric called, Fault Trigger Metric (FTM). The value of FTM is calculated at event, module and system level and can form the basis to predict reliability/testing effort of the web application. The value of FTM at system level is called Fault Vulnerability Metric (FVM) and can form the basis to predict reliability/testing effort of the web application. Systems with high FVM value will be less reliable and hence will require more testing effort.

*Key words*: fault tree, event, fault, web application, reliability, testing effort
*Communicated by*: M. Gaedke & Q. Li

## 1 Introduction

As we move towards pervasive computing, web applications have gained importance in many application domains. As web applications are used in real time applications, their reliability becomes important. One of the challenges in testing web application is the enormous interaction of events invoked by the users. The large number of interactions with a web application results in a large number of test cases, making testing of web application difficult. In a web application, a system is executed in large number of client configurations, making space of possible configurations very large. This requires extensive testing of a web application. A web application consists of a number of web pages interconnected by static links or dynamic links. Static links also known as HTML links or hyperlinks are connections from one web page or web resource to another. A static link is an element, an image or a text that users can click on and navigate to the next web page. The default behavior of static link is retrieval of another web page which is obtained by selecting the link. On the other hand dynamic links carry out user session dependent actions [9].

The complexity of a system is closely related with maintainability, testing efforts, and understandability [4]. Growing complexity of web applications has therefore led to high maintenance cost. In the past, several researchers have proposed metrics for predicting the maintainability of web applications. Ghosheh, E. et al. [2] have proposed metrics using Web Application Extension (WAE) for Unified Modeling Language (UML) class diagrams to measure the maintainability of the web applications. Jung, W. et al. [4] have proposed a complexity measure for a web application which is related to the maintenance efforts. Mao, C. [6] has proposed data complexity metric using def-use pairs in order to understand the system and to do its defect analysis. Marchetto, A. et al. [7] have proposed a quality model based on object oriented metrics to describe the structural properties of the web based systems and to analyze them. Thi, Q. P. et al. [11] have proposed a suite of complexity metrics using web service definition language in order to evaluate the web services. Zhang, Y. et al. [19] proposed a set of web site complexity metrics using only the number of static links. Shahzad, A. et al. [9] generated test cases from web navigation graph by considering dynamic links. Butkiewicz, M. et al. [22] identified a set of metrics to characterize the complexity of websites, both at a content level and service level. Panda, S.K. et al. [23] measured the website usability, quality, complexity metrics based on the construction of hierarchical structure of a web site, digit of strikeouts, and cyclomatic complexity of a web site roadmap.

One of the popular techniques used in industries for ensuring safety, risk and reliability analysis of a system is FTA [13, 17]. A fault tree is a logical diagram used to identify the critical events that cause the system to fail [8, 3]. FTA is a qualitative model that provides information on the causes of undesired events [14]. It is a deductive approach, providing useful information on the causes of undesired events [10]. For example, for an undesired event (say server failure) of a system, the causes of undesired events are deduced using a systematic backward stepping process [10]. FTA can be applied to both an existing system and to a system that is being designed [10]. When applied to a system being designed, safety requirements for the system can be derived from the fault trees. The failure probability of the components can be estimated from the fault trees [10]. FTA can also be used for the development of performance based critical systems [10]. When applied to existing system, FTA can be used to identify weaknesses, to evaluate the possible newer version of the system and to monitor and predict the behavior of a system [10]. FTA is typically used for hardware systems, but the recent contribution shows its wide use in software as well. The UML statechart diagram and sequence diagram are transformed into a corresponding software fault tree, enabling the testing team to identify modules that could affect the safety of the system at the design stage [13]. Helmer, G. et al. [3] discussed software fault tree for its use in requirement identification and analysis in an Intrusion Detection System. The safety requirements for the systems are derived from the software fault tree. Software fault trees are closely related to threat trees [3]. Threat trees are notable tools in the security analysis process called “threat modeling” [33]. Needham, D. et al. [8] proposed the key node safety metric for identifying “key nodes” within a fault tree for measuring the safety of the system. A key node is defined as a node in a fault tree that allows a failure to propagate towards the tree root if and only if multiple failure conditions exist in the node [8]. Wang, Y. et al. [16] have proposed a computer-aided fault tree synthesis methodology to construct a fault tree automatically so as to mitigate the risk involved in the conventional manual construction of fault tree. Ying, R. et al. [18] used FTA to improve the efficiency of software testing. The system failure model is constructed using fault tree. The qualitative analysis of FTA is done in order to construct minimal cut sets using down law. In down law, one by one top-down processing is done, starting from the top event. Finally, all the resulting cut sets are compared to derive minimal cut sets. A cut set is defined as a set of basic events whose occurrence causes the system to fail [12]. Tian, P. et al. [12] used FTA for determining the software module reliability and using genetic algorithm, software reliability allocation model is optimized. Leveson, N. G. [5] emphasized on using the results of software fault tree safety analysis for identifying safety constraints that must be met by the software's requirements [8].

In this paper a metric called Fault Trigger Metric (FTM) has been proposed for web based systems to predict their reliability. FTM is computed using FTA technique and web links (static link and dynamic link). FTA technique has been used to identify undesired events of a web application. Using top-down approach, the web application is divided into component modules and a fault tree is drawn for each component module of web application. Using our proposed approach, a numerical measure called Relative Information Flow (RIF) is assigned to each fault event at a leaf node in a fault tree. RIF is calculated using the number of static links or dynamic links connecting web pages in a web application. The numerical measures are thereafter used to compute FTM at event, module and system level. The FTM of a module for example, is an indicator of the probability of triggering a fault due to that component module in software application. This helps in identifying the modules which make software application more vulnerable to failures. Therefore, the modules with higher values of the FTM can be redesigned even before the actual testing of web application begins. This saves the testing effort and makes the web based system resilient to faults and henceforth more reliable. We will call FTM of the system Fault Vulnerability Metric (FVM). FVM is the measure of system vulnerability to undesired events or faults in web application and is based on probability of faults triggered by different modules in web application.

The organization of the paper is as follows. The related work is presented in section 2. Section 3 illustrates the basic concepts underlying the fault tree construction. Section 4 illustrates our proposed approach. The two case studies illustrating our approach are presented in section 5. Finally, section 6 concludes our research work and also highlights the future work.

## 2 Related Work

Web applications have gained remarkable significance in many application domains. Due to an overwhelming increase in size and complexity, measuring the reliability/testing effort of web applications has become a challenging issue and a focus of research. Several researchers have contributed their research in the field of web based systems by modeling navigation behavior of web applications, measuring the important factors like reliability, testability and complexity of web applications using different techniques.

In 1999, Chong, C.W. et al. [25] analyzed the current user's surfing behavior and proposed efficient pre-order and Hamiltonian navigation models for increasing naive users surfing efficiency. Hamiltonian surfing is defined as the web navigation process where the users access each web site within the same domain exactly once [25]. Pre-order surfing is defined as a depth-first-surfing that is useful to naive users who are interested in hierarchical surfing [25]. A fuzzy distance measure has also been used to determine the optimal path leading to a specific target page for the experienced users. In 2000, Leung, K.R.P.H., et al. [26] proposed a web navigation model based on statechart for modeling complex and dynamic behavior of web applications. A web navigation model is proposed to model intra page navigation, inter page navigation, frame based navigation and dynamic content on the client side and server side respectively. Sabharwal, S. et al. [32] proposed a graph based modeling technique to model the navigation behavior of a web application for the purpose of testing from the information extracted from the requirement and design documents of the web application.

Mendes, E. et al. [27] used an undergraduate university course as a case study and collected metrics corresponding to web applications, developers, and tools. These metrics were used to generate models for predicting design and authoring effort for future web applications. Marchetto, A. et al. [7] defined a quality model for web applications based on a metrics suite to help the user define a quantitative system to measure web software and to analyze/predict quality factors via structural properties. These metrics are useful to measure some important attributes, such as complexity, coupling, size, cohesion, reliability, defect density, and so on [7]. Dhawan, S. et al. [28] identified the web metrics for detecting and resolving important issues of web based applications like poor reliability, long response time and other important issues. An approach was introduced for determining the reliability, usability index, error estimating function, web replacement policies, user model behavior model graph, Rest Style (RS) web application effort estimation and effort estimation for web-based systems. Alagappan, B. et al. [29] proposed web metrics based on its usability and effectiveness for different web domain users of the web page. The proposed class of web metrics which encompasses the structure, visual appearance and the interactive behavior of any web page are represented as utility oriented metrics, quality oriented metrics, behavior oriented metrics and visual oriented metrics. Butkiewicz, M. et al. [22] identified a set of metrics to characterize the complexity of web sites. The complexity of web pages is characterized on the basis of content they include and the services they offer. A web page is characterized by the content fetched in rendering it like the number of objects fetched, the sizes of these objects, and the types of content [22].

## 3 Theoretical Foundations

In this section we will briefly introduce fault trees, statechart diagrams and different types of links in a web application. FTA is described as an analytic technique whereby an undesired state is specified and the system is analyzed in context of environment and operations to find all credible ways in which an undesired event can occur [10, 14]. FTA is also one of the methods under study at the Software Assurance Technology Center (SATC) at NASA’s Goddard Space Flight Center [30]. The purpose is to determine its relevance to reduce the risks of software and to identify commercial software tools to assist in using this technique [30]. FTA can be described as a deductive approach for determining the failure probability of software components or a system.

A fault tree is a logical diagram of basic failures called “Primary failures” [15]. The top failure (root) in fault tree is an undesired event that causes system failure. The top failure is represented as the union of node failures. The ‘AND’ gate and ‘OR’ gate is used to construct a fault tree. A sample fault tree is shown in Figure 1. The rectangle represents fault events. A rectangle symbol at the top of a fault tree represents failure or top event. Rectangle symbols below top event are fault events/causative events causing a top event or a failure to occur. A circle below fault event represents a basic event in a fault tree. There are no gates or events below basic event. Basic events require no further development and are found on the bottom of the tree. A diamond represents undeveloped or non primal event. Non primal events are not fully developed because of lack of information or significance. A fault tree can have non primal events at the bottom. The triangle or transfer symbol signifies a connection to another fault tree. A fault tree branch below the transfer symbol is transferred to another location in a fault tree. Letters, numbers or figures are used to identify sets of transfer symbol in a fault tree. Few relevant symbols used in a fault tree are shown in Figure 2.

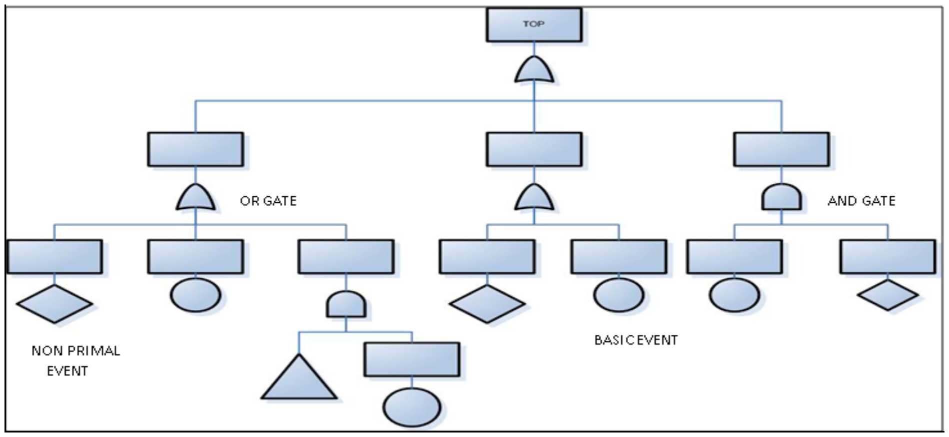


Figure 1 Fault Tree

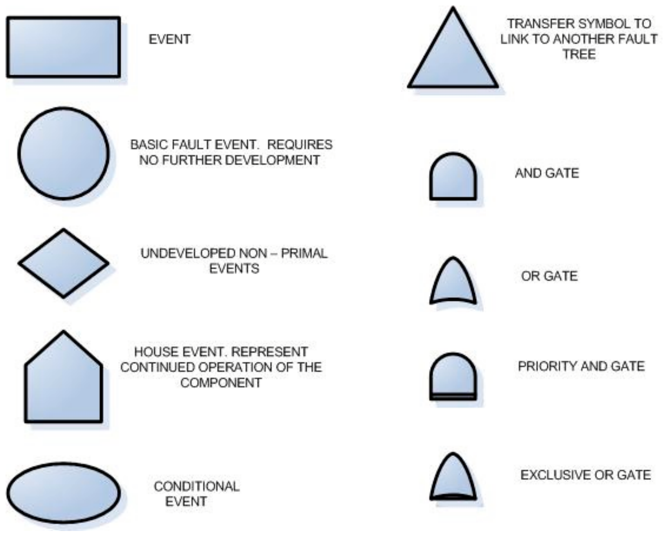

EVENT
TRANSFER SYMBOL TO LINK TO ANOTHER FAULT TREE
BASIC FAULT EVENT. REQUIRES NO FURTHER DEVELOPMENT
AND GATE
UNDEVELOPED NON – PRIMAL EVENTS
OR GATE
HOUSE EVENT. REPRESENT CONTINUED OPERATION OF THE COMPONENT
PRIORITY AND GATE
CONDITIONAL EVENT
EXCLUSIVE OR GATE

Figure 2 Fault Tree Symbols (Vesely et al., 1981)

UML statechart diagram is an extension of the Finite State Machine, showing a state machine with states, transitions, events and actions [20]. Statechart diagrams mainly express the information of the state transitions and actions response for a finite state system according to external events inspires [20, 21]. UML statechart diagrams are used to show the states an object can have during its lifetime. A transition is a relationship between two states, indicating a possible change from one state to another [20]. The events in statechart diagram cause the state to change along with its responses. Actions are performed in response to events received while the object is in the state, without changing the state.

According to Shahzad, A. et al. [9], a web application typically consists of interconnected static web pages or dynamic web pages. The interconnection is primarily defined by static links (i.e. Hyperlinks), dynamic links (i.e. that carry out user session dependent actions) or by button actions (i.e. Form submission, etc.) [9]. A static web page (e.g., HTML) is a client side page that is delivered to the user exactly as stored. According to Shahzad et al. [9], a dynamic link dependent on session data would behave differently if the user is logged in or logged off. According to Leung, K.R.P.H., et al. [26], a static web page is defined as a web page that retains the same HTML for all the client requests of the same URL. It must also contain no reactive or executable components [26]. A dynamic web page is defined as a web page that returns different HTML for client requests of the same URL or contains reactive or executable components [26].

## 4 Proposed Approach

We propose a top-down strategy for problem solving. The web application is divided into component modules. Thereafter, a fault tree is constructed for each module of the web application. The potential undesired events of the modules are determined and their basic causes are resolved using FTA. Finally the FTM/vulnerability of the web application to failure is computed using the procedure outlined below:

*Procedure*

1. *Draw a UML statechart diagram for each module of the web application* to detail every event that triggers the transition.

2. *Identify undesired events in the modules of the web application.*

3. *Identify the basic causes leading to undesired events in the web application.*

4. *Transform a statechart diagram into a fault tree*. The undesired events causing failures in the system are the top events in a fault tree. Figure 3 shows a fault tree where $E_1$ is a top event. Causative events are laid out in a fault tree with branches connected by logic gates performing logical functions [1]. An intermediate event is a fault event which occurs because of one or more antecedent causes acting through logic gates [14]. All intermediate events are symbolized by rectangles in a fault tree [14]. In Figure 3, events $E_2$, $E_3$, $E_4$ and $E_5$ are fault events/causative events of top event $E_1$ where $E_3$ represents an intermediate event, $E_2$ and $E_4$ represent basic events and $E_5$ represent non-primal event.

5. *Generate navigation graph of web application*. A web navigation graph is a simple graph denoted by a set of edges and vertices where each edge represents a static link or dynamic link and each vertex represents a web page.

   The usage scenario of web application is captured using web navigation graph constructed using Microsoft Visio 2003 tool**.** The web navigation graph is generated by using the Interactive Hyperlink Selection feature in Microsoft Visio 2003 tool. Edges connecting the nodes in a web navigation graph represent either a static or a dynamic link. Microsoft Visio 2003 tool determines the number of edges connected to a vertex irrespective of the nature of the link. In our work, cycle in a web navigation graph is traversed only once.

   Suppose event $E_4$ in Figure 3 represents a fault event 'User Forgot Password'. A web navigation graph generated using Microsoft Visio 2003 tool for a fault event 'User Forgot Password' is shown in Figure 4. Figure 4 shows count of number of number of links to $E_4$ and a number of links from $E_4$. By selecting the custom properties in Microsoft Visio 2003 tool for $E_4$, the number of links (static link and dynamic link) input to $E_4$ is 1 and the number of links (static link or dynamic link) output from $E_4$ is 32.

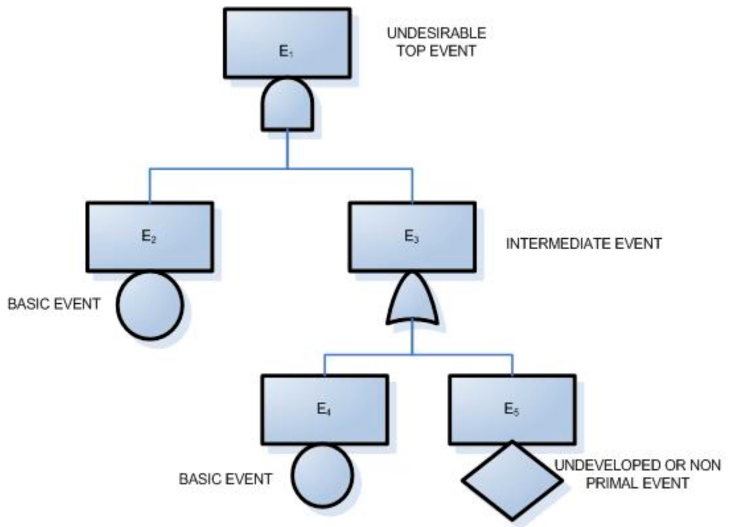
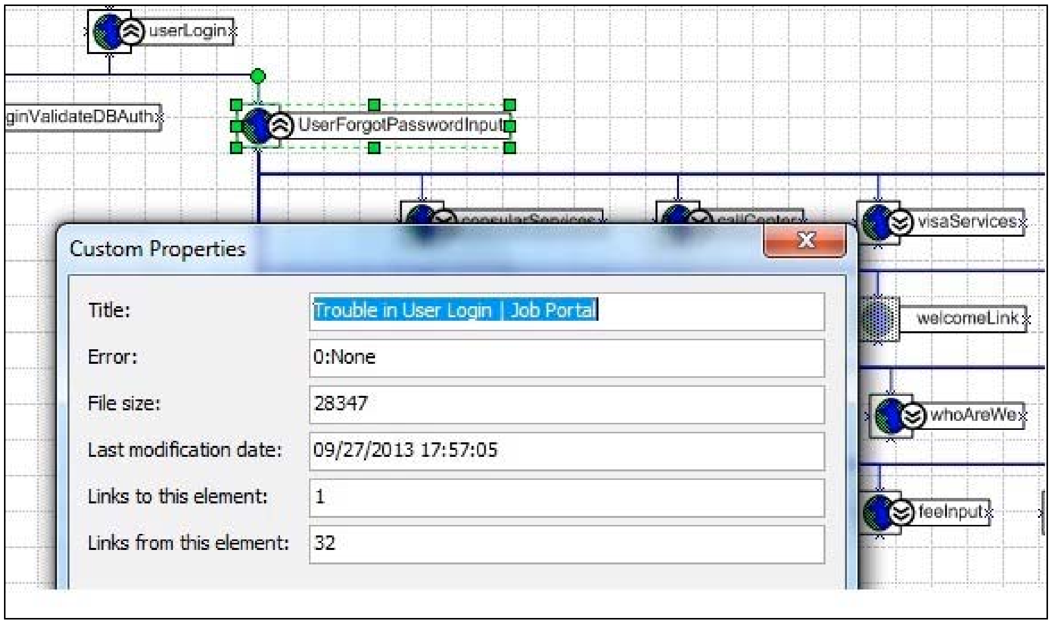


Figure 3 Fault Tree Example for Module of Web Application



| | |
|---|---|
| Title: | Trouble in User Login \| Job Portal |
| Error: | 0:None |
| File size: | 28347 |
| Last modification date: | 09/27/2013 17:57:05 |
| Links to this element: | 1 |
| Links from this element: | 32 |

Figure 4 Navigation Graph of a Fault Event 'User Forgot Password' or $E_4$

6. *Apply proposed metrics for determining FTM of the system/vulnerability of web application to failure.* The procedure for determining FTM of the system is as follows:

a) Calculate RIF of each causative event at a leaf node in a fault tree.

To calculate RIF, we need to identify the number of links (edges) input to and output from each causative event at a leaf node. The aim is to determine the probability of triggering a fault due to each causative event at a leaf node in a software application.

In our work, we have adopted the Information Flow Model [24] proposed by Henry and Kafura for determining the Information Flow (IF) of each causative event in a fault tree. Henry and Kafura defined the structure complexity for a procedure or component A as:-

$$\text{IF (A)} = (\text{FAN-IN (A)} * \text{FAN-OUT (A)})^2 \tag{1}$$

where FAN-IN (A) is a count of the number of other components that can call or pass control to component A and FAN-OUT (A) is a number of components that are called by component A. In our work, we have considered FAN-IN (A) as a count of the number of other web pages that can call or pass control to web page A via a static link or dynamic link and FAN-OUT (A) is a number of web pages that are called from web page A via a static link or dynamic link.

Initially, the IF of each causative event at a leaf node in a fault tree is determined from the navigation graph of web application generated in step 5 using Microsoft Visio 2003 tool. The IF of a causative event at leaf node, $E_i$ (where i = 1, 2, 3...n and n is number of events) in a fault tree is computed by multiplying the count of the number of static or dynamic links, $c_i$ input to $E_i$ and count of the number of static or dynamic links, $c_o$ output from $E_i$. The count of the number of input and output links on a web page is obtained from the navigation graph of the web application. To avoid large computational value, we have removed the square function in equation 1. Therefore, IF of each causative event $E_i$ at a leaf node in a fault tree is determined by applying equation 2.

$$\text{IF}(E_i) = (\text{Number of } c_i \text{ to } E_i * \text{Number of } c_o \text{ from } E_i) \tag{2}$$

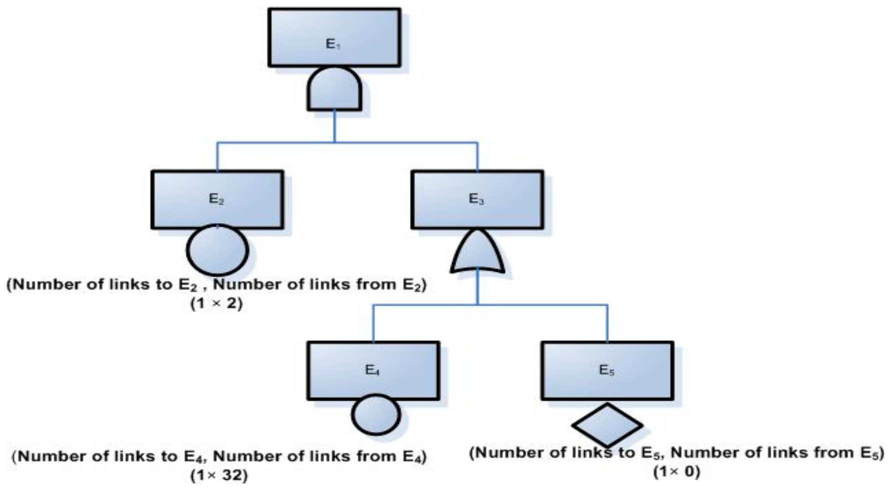

E₁
E₂
E₃
(Number of links to E₂ , Number of links from E₂)
(1 × 2)
E₄
E₅
(Number of links to E₄, Number of links from E₄)
(1× 32)
(Number of links to E₅, Number of links from E₅)
(1× 0)

Figure 5 IF of Causative Events

In Figure 5, fault tree shows the IF of each event $E_i$ determined from navigation graph of web application captured using Microsoft Visio 2003 tool. Next, Total Information Flow (TIF) of a fault tree FT corresponding to the module of web application is calculated by applying equation 3.

$$TIF(FT) = \sum_{i=1}^{n} (IF(E_i)) \tag{3}$$

$\text{IF}(E_i)$ represents IF of event $E_i$ at leaf node in a fault tree. For example, in Figure 5 TIF of fault tree FT calculated using equation 3 will be:-

$$\begin{aligned} \text{TIF (FT)} &= \text{IF}(E_2) + \text{IF}(E_3) = \text{IF}(E_2) + \text{IF}(E_4) + \text{IF}(E_5) \\ &= ((1* 2) + (1 * 32) + (1 * 0)) = 34 \end{aligned}$$

RIF of each causative event $E_i$ at leaf node in a fault tree can now be computed using equation 4 as given below.

$$RIF(E_i) = \frac{IF(E_i)}{TIF} \tag{4}$$

For example, in Figure 5, RIF of causative events at leaf node using equation 4 will be:-

$$RIF(E_2) = (1*2) / 34 = 0.06$$

$$RIF(E_4) = (1*32) / 34 = 0.94$$

$$RIF(E_5) = (1*0) / 34 = 0$$

b) Assign Strength (G) to gates used in a fault tree based on their probability of triggering a fault.
Causative events in a fault tree are connected by logic gates. The probability of triggering a fault by a causative event in a fault tree also depends on the type of logic gate attached to it. To determine the value of FTM at module level, we therefore need to calculate the strength of logic gates in a fault tree.

Table 1 Truth Table of Fault Tree Logic Gates

| **INPUTS** | | **OUTPUT** | | | | | |
|---|---|---|---|---|---|---|---|
| A | B | AND | PAND | NOR | OR | NAND | XOR |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

Table 2 Strength of Fault Tree Logic Gates

| **S. No.** | **Gate** | **Strength** |
|---|---|---|
| 1. | AND, PAND, NOR | 1 |
| 2. | OR, NAND | $2^n$-1 |
| 3. | XOR | n |

The strength of a particular gate is derived from its truth table. The commonly used gates in a fault tree are OR, XOR, AND, NOR, NAND and priority AND (PAND) gate. The truth tables of these gates are shown in Table 1. The truth table of logic gates has two inputs A and B. In truth table, "0" represents false value and "1" represents true value. The strength assigned to various gates is shown in Table 2. In AND gate, output or undesired event occurs if and only if all incoming inputs or causative events occur. So, it has a low probability of causing undesired event and is assigned strength of value 1. In PAND gate, the output event occurs when both input events get executed and one input event precedes another. The input events must be ordered in PAND gate. The AND and PAND gates have similar functionality except ordered or prioritized inputs in PAND gate. Thus, PAND gate has been assigned the same strength as that of AND gate, i.e., 1. The NOR gate functions like a combination of an OR gate and a NOT gate. The NOR gate is used to indicate that the output occurs when all the input events are absent and is assigned the same strength as that of AND and PAND gates, i.e., 1. In OR gates, output or undesired event occurs if and only if at least one of the input gets executed. Therefore, if n is the number of causative events, then the probability of triggering a fault by at least one of the input out of n causative events is $2^n$-1. Therefore, strength assigned to OR gate is $2^n$-1. In NAND gate, output event occurs when at least one of the input events is absent. The probability of triggering a fault by NAND gate is same as OR gates. Therefore, NAND gate is assigned same strength as that of OR gate, i.e., $2^n$-1. In XOR gate, output event occurs if exactly one of the two input events occurs and the other input event does not occur. So, the strength of XOR gate is n.

c) Compute the value of *FTM* of each intermediate event $E_i$ in a fault tree.

An intermediate event occurs because of one or more antecedent causes acting through logic gates [14]. In FTA, the causes of undesired events are deduced using systematic backward stepping process (bottom to top). Therefore, the value of FTM of a module is deduced by determining the FTM of intermediate events at the lowest level first.

In a fault tree, FTM is computed on the basis of strength assigned to the logic gate Strength (G) attached to an event and the RIF value of causative events connected to the logic gate. It is defined as a measure of probability of triggering a fault in the module due to an intermediate event. The algorithm for calculating the value of FTM for an intermediate event FTM ($E_i$) in a fault tree is shown in Figure 6. For example, for a fault tree given in Figure 5, FTM for event $E_3$, i.e., FTM ($E_3$) is calculated as given below:-

$$\begin{aligned}\text{FTM}(E_3) &= ((\text{RIF}(E_4) + \text{RIF}(E_5)) \times \text{Strength}(G)) \\ &= ((\text{RIF}(E_4) + \text{RIF}(E_5)) \times 2^2 - 1) \\ &= (0.94 + 0) \times 3 = 2.82\end{aligned}$$

**Algorithm 1: Compute FTM of an Intermediate Event: Compute_FTM (E)**

```
   Let $E_i$ is an intermediate event of a fault tree.
   RIF ($E_i$) is Relative Information Flow of event $E_i$.
   $E_{ij}$ is the $j^{th}$ causative event of $E_i$ triggering the event $E_i$ connected through gate G (where j =
1...n).
   Strength (G) is the probability of triggering a fault by the logic gate, G rooted at $E_i$.
   Let FTM ($E_i$) represents the Fault Trigger Metric value of event $E_i$.
   n is number of causative events of event $E_i$
for each event $E_i$
    1.     Set FTM ($E_i$) = 0
    2.     for ( j = 1 to n)
    3.         FTM ($E_i$) = FTM ($E_i$) + RIF ($E_{ij}$)
    4.         FTM($E_i$) = FTM($E_i$) * Strength(G)
    5.     End for
```

Figure 6 Algorithm for Computing FTM of Intermediate Event

d) Compute the value of FTM of each fault tree FTM (FT) in web application.

To compute the value of FTM at module level, we need to determine the RIF value of the causative event triggering a top event in a fault tree and strength of logic gate attached to top event in a fault tree.

The FTM value of each fault tree is determined by computing the FTM value of a top event using equation 5. FTM (FT) for a module gives a measure of probability of triggering a fault in web application due to that module.

$$FTM(FT) = \sum_{i=1}^{n} (RIF(E_i)) * Strength(G_{top}) \quad (5)$$

Where $E_i$ represent a causative event triggering a top event of fault tree FT connected through logic gate $G_{top}$. Strength ($G_{top}$) is the probability of triggering a fault by gate $G_{top}$. The higher is the probability of FTM (FT) for a module, higher will be the probability of occurrence of fault in web application due to that module.

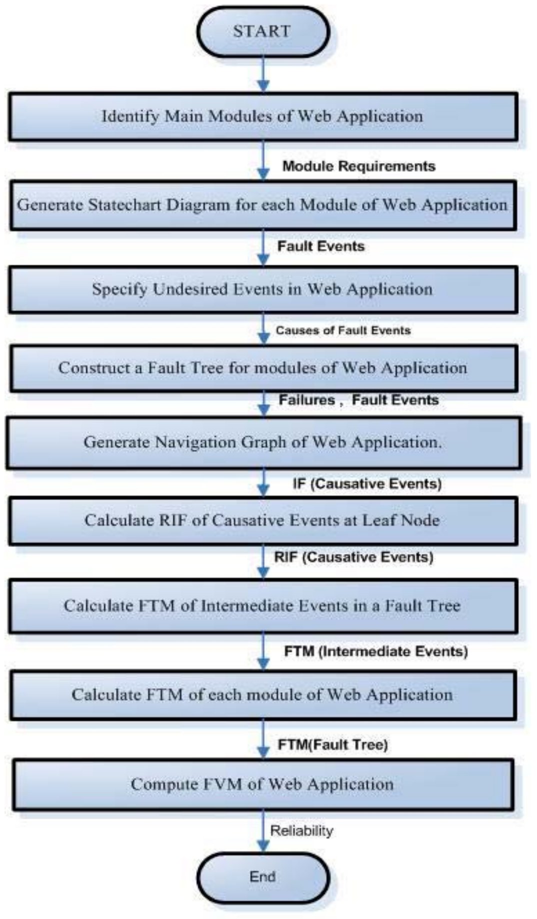

START
Identify Main Modules of Web Application
Module Requirements
Generate Statechart Diagram for each Module of Web Application
Fault Events
Specify Undesired Events in Web Application
Causes of Fault Events
Construct a Fault Tree for modules of Web Application
Failures , Fault Events
Generate Navigation Graph of Web Application.
IF (Causative Events)
Calculate RIF of Causative Events at Leaf Node
RIF (Causative Events)
Calculate FTM of Intermediate Events in a Fault Tree
FTM (Intermediate Events)
Calculate FTM of each module of Web Application
FTM(Fault Tree)
Compute FVM of Web Application
Reliability
End

Figure 7 Flowchart for the FVM Determination of Web Application

e) Compute the value of FTM (SYS) or FVM of complete web application.

To determine the reliability of web application, the value of FTM at system level is computed. To compute the value of FTM at system level or FVM, we require the value of FTM (FT) or FTM at module level. FTM (SYS) or FVM is computed by summing up the FTM value of all the modules of the web application as given in equation 6.

$$FTM(SYS) = \sum_{j=1}^{m} FTM(FT_j) \quad (6)$$

The higher is the value of FTM (SYS) or FVM, lower will be the reliability of the system which in turn will require more testing effort. The flow chart of our proposed approach is shown in Figure 7.

## 5 Determining Reliability of Web based Systems

In this section, we will discuss our proposed approach for determining reliability/testing effort of web based systems by taking two case studies. As already discussed in the previous sections, FTA technique has been used to analyze the potential faults of a web application. To estimate FVM of the system, the potential faults of the web application are identified and laid out in a fault tree in top to bottom manner. The FVM is further used to estimate the reliability of the web application. A web based system with a higher value of FVM is more vulnerable to faults and therefore less reliable and vice versa. Also a less reliable system will require more testing effort which is not preferable.

### 5.1. *Case Study 1: Web Based Job Portal Application*

In this section, we present a case study wherein we will compute the FVM of a web based Job Portal application using our proposed approach. The steps for determining the FVM value of web application using our approach are as follows:

#### *5.1.1. System Analysis*

To illustrate our approach, we have gathered requirements for a web based Job Portal. The goal of the application is to provide a portal for job seekers to submit their curriculum vitae (CV). The employer of the company can shortlist best CV based on the criteria posted for a vacancy. The application requires the employee and Jobseeker registration on the Job Portal. This requires a new user ID (alphanumeric) and password (minimum length 6) for the registration. There is a separate control panel for employee and job seeker handling the web application. The employer can search jobseekers profile by keywords, download CV, job profile and date of job posted, can retrieve passwords reset by automated generated mail, can register employer or a company and post jobs. Job seeker can search jobs based on criteria (experience, qualification, etc.), add and update own CV andget new job notification by mail. The application has browser compatibility with Mozilla, email is verified when user register on the web site.

Using a top-down strategy, the application is divided into three modules, i.e. Employee Login, Job Seeker Login and Search a Job. The statechart diagram for each module is shown in Figure 8, Figure 9 and Figure 10 respectively.

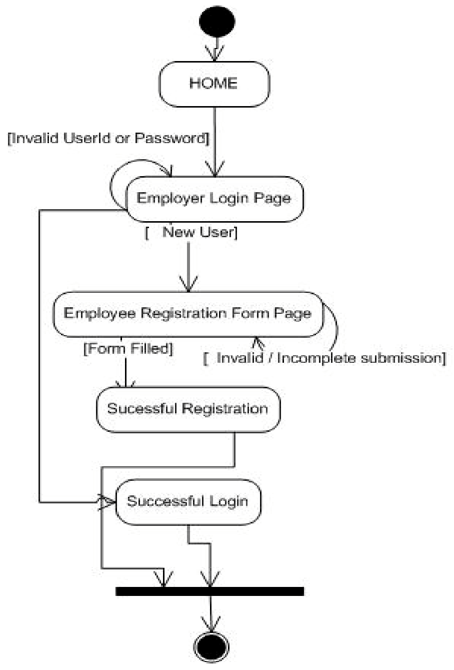

HOME
[Invalid UserId or Password]
Employer Login Page
[ New User]
Employee Registration Form Page
[Form Filled]
[ Invalid / Incomplete submission]
Sucessful Registration
Successful Login

Figure 8 Statechart Diagram for Employee Login

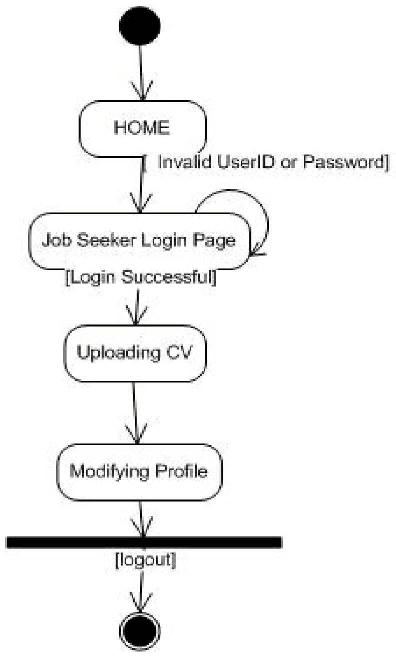


Figure 9 Statechart Diagram for Job Seeker Login

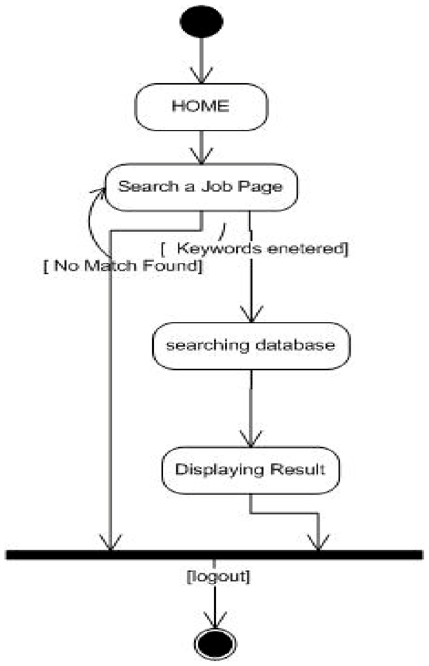


Figure 10 Statechart Diagram for Search a Job

### 5.1.2. *Fault Tree for Job Portal Web Application*

Fault trees are drawn for different modules of the web application. The fault trees are constructed by identifying the fault events and determining the relationship between them for causing the failure event in the particular module.

Table 3 Failure Modes and Mechanism of Employee Login/Jobseeker Login Module

| Failure Effect | Failure Mode | Mechanism |
|---|---|---|
| Login Failed | • Invalid User Id<br>• Invalid Password<br>• Incompatible Browser<br>• Server connection Failed | Human error, expired account<br>Human error<br>Browser used other than Mozilla<br>Network error |

Table 4 Failures Modes and Mechanism of Search a Job Module

| Failure Effect | Failure Mode | Mechanism |
|---|---|---|
| Job Not Found | • Page Failed to Load<br>• No Match Found | Browser used other than Mozilla, Server load Increases<br>The keyword does not exist in the database |

The system is analyzed and the top events or undesired events in web application are defined. The faults identified in Job Portal application are shown in Table 3 and Table 4 respectively. Fault tree is constructed by determining the relationship between the causes of top event and identifying them to resolve the failure.

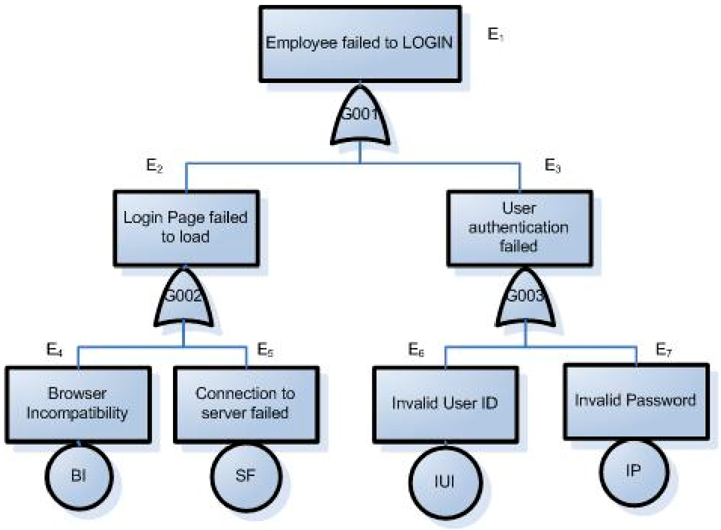


Figure 11 Fault Tree for Employee Login

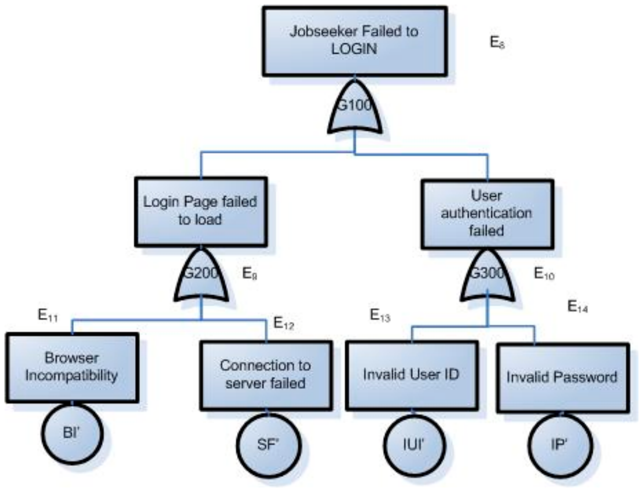

Jobseeker Failed to LOGIN
E8
G100
Login Page failed to load
User authentication failed
G200
E9
G300
E10
E11
E12
E13
E14
Browser Incompatibility
Connection to server failed
Invalid User ID
Invalid Password
BI'
SF'
IUI'
IP'

Figure 12 Fault Tree for Job Seeker Login

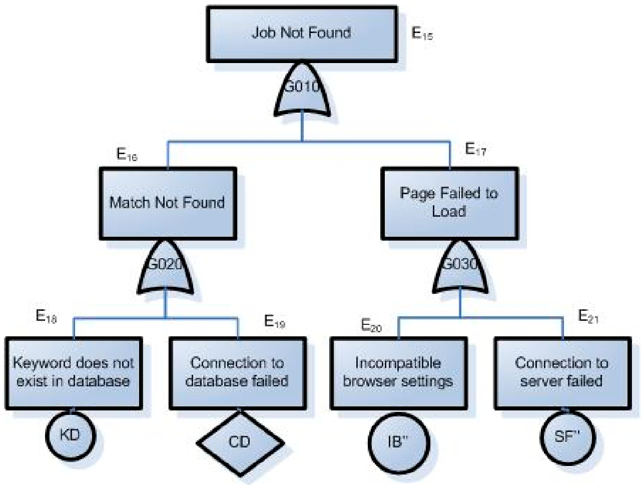

Job Not Found
E15
G010
E16
E17
Match Not Found
Page Failed to Load
G020
G030
E18
E19
E20
E21
Keyword does not exist in database
Connection to database failed
Incompatible browser settings
Connection to server failed
KD
CD
IB''
SF''

Figure 13 Fault Tree for Search a Job

In Figure 11, $E_1$ is a top event, $E_2$ and $E_3$ are intermediate events/causative event where as $E_4$, $E_5$, $E_6$ and $E_7$ are basic/causative events. In Figure 12, $E_8$ is a top event, $E_9$ and $E_{10}$ are intermediate events/causative events, $E_{11}$, $E_{12}$, $E_{13}$ and $E_{14}$ are basic events/causative events. In Figure 13, $E_{15}$ is a top event, $E_{16}$ and $E_{17}$ are intermediate events/basic events, $E_{18}$, $E_{20}$, $E_{21}$ are basic events/causative events and $E_{19}$ is non-primal event/causative event.

### 5.1.3. *Determining Reliability of Job Portal Web Application*

Once the fault trees are constructed, next we draw the web navigation graph of the Job Portal application using Microsoft Visio Studio 2003 tool as shown in Figure 14.

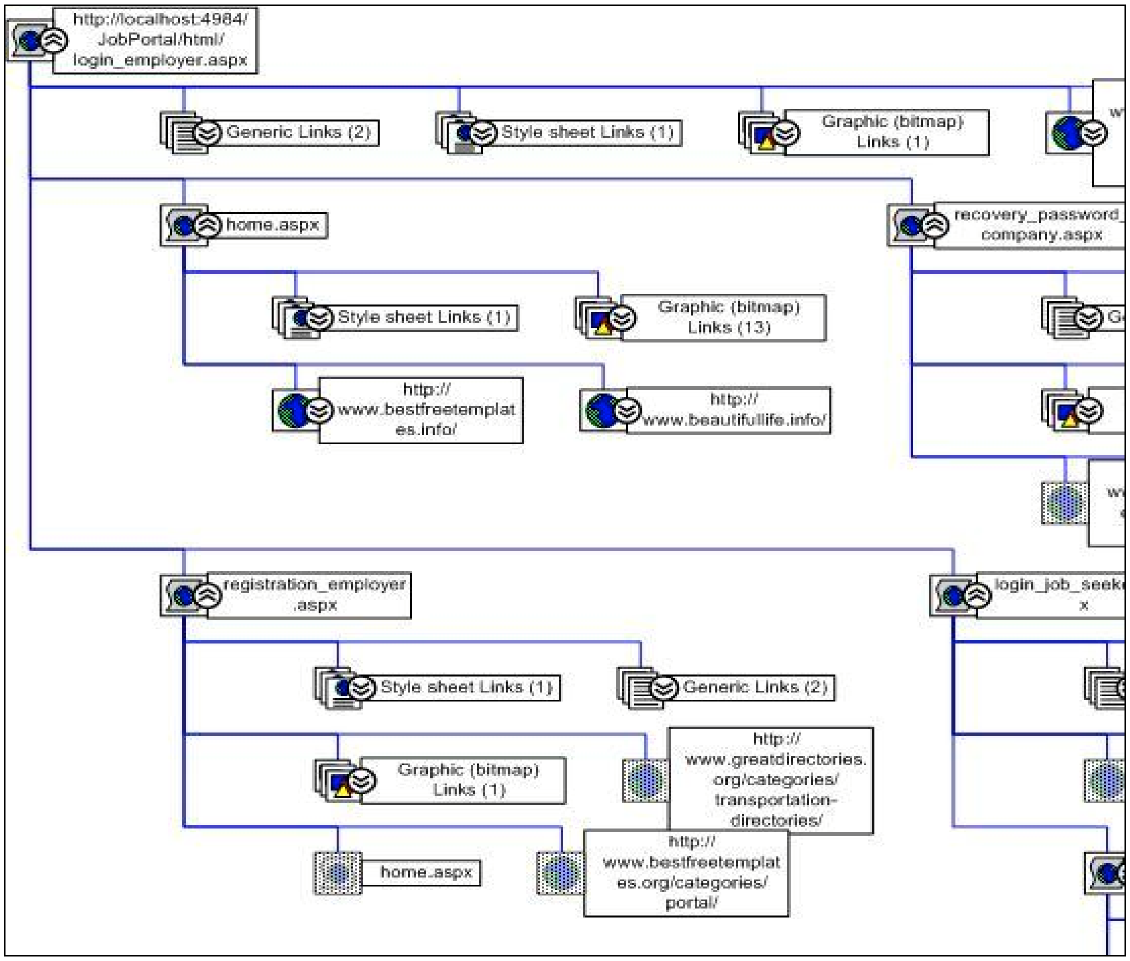

http://localhost:4984/JobPortal/html/login_employer.aspx
Generic Links (2)
Style sheet Links (1)
Graphic (bitmap) Links (1)
home.aspx
recovery_password_company.aspx
Style sheet Links (1)
Graphic (bitmap) Links (13)
http://www.bestfreetemplates.info/
http://www.beautifullife.info/
registration_employer.aspx
login_job_seek x
Style sheet Links (1)
Generic Links (2)
Graphic (bitmap) Links (1)
http://www.greatdirectories.org/categories/transportation-directories/
home.aspx
http://www.bestfreetemplates.org/categories/portal/

Figure 14 Web Page Navigation Graph of Job Portal Web Application

### *5.1.3.1. Applying Proposed Measures on Fault Tree*

The metrics proposed in section 4 are applied to the fault tree constructed for Job Portal application. The web page navigation graph as shown in Figure 14 represents the lower level view of our application. We have applied proposed metrics on the fault tree constructed for three modules, namely Employee Login, Job Seeker Login and Search a Job as shown in Figure 11, Figure 12 and Figure 13 respectively. The steps for determining FVM of Job Portal application are illustrated in Table 5.

The RIF of each causative event in a fault tree is determined by applying equation 4. The FTM value of all intermediate events in fault trees corresponding to the modules of the Job Portal web application is calculated by applying Algorithm 1 given in Figure 6. The FTM value of each fault tree in Job Portal application is calculated by applying equation 5. Finally the FVM value of the Job Portal system, FVM ($SYS_1$) is determined by applying equation 6 as shown below:

$$FVM\ (SYS_1) = 45.5 + 44.4 + 45 = 134.9$$

Table 5 Calculation of FTM of different modules of Job Portal Application

| | MODULE | | |
|---|---|---|---|
| **Metric** | **Employee Login ($FT_1$)** | **Job Seeker ($FT_2$)** | **Search a Job ($FT_3$)** |
| IF ($E_i$) | IF($E_4$) = 1*1 = 1<br>IF($E_5$) = 1*6 = 6<br>IF($E_6$) = 1*6 = 6<br>IF($E_7$) = 1*6 = 6 | IF ($E_{11}$) = 1*1 = 1<br>IF ($E_{12}$) = 1*1 = 1<br>IF ($E_{13}$) = 1*32 = 32<br>IF ($E_{14}$) = 1*32 = 32 | IF ($E_{18}$) = 1*7 = 7<br>IF ($E_{19}$) = 1*1 = 1<br>IF ($E_{20}$) = 1*1 = 1<br>IF ($E_{21}$) = 1*1 = 1 |
| TIF ($FT_i$) | TIF ($FT_1$) = 19 | TIF ($FT_2$) = 66 | TIF ($FT_3$) = 10 |
| RIF ($E_i$) | RIF ($E_4$) = 1/19 = 0.05<br>RIF ($E_5$) = 6/19 = 0.32<br>RIF ($E_6$) = 6/19 = 0.32<br>RIF ($E_7$) = 6/19 = 0.32<br>RIF ($E_4$+$E_5$+$E_6$+$E_7$) = 1.01 | RIF ($E_{11}$) = 1/66 = 0.01<br>RIF ($E_{12}$) =1/66 = 0.01<br>RIF ($E_{13}$) = 32/66 = 0.48<br>RIF ($E_{14}$) = 32/66 = 0.48<br>RIF ($E_{11}$+$E_{12}$+$E_{13}$+$E_{14}$) = 0.98 | RIF ($E_{18}$) = 7 / 10 = 0.7<br>RIF ($E_{19}$) = 1/ 10 = 0.1<br>RIF ($E_{20}$) = 1/ 10 = 0.1<br>RIF ($E_{21}$) = 1/ 10 = 0.1<br>RIF ($E_{18}$+$E_{19}$+ $E_{20}$+ $E_{21}$) = 1.0 |
| FTM ($E_i$) | FTM ($E_2$) = (0.05+0.32) * ($2^2$-1) = 1.11<br>FTM ($E_3$) = (0.32 + 0.32) * ($2^2$-1) =1.92 | FTM ($E_9$) =<br>(0.01+0.01) * ($2^2$-1) = 0.06<br>FTM ($E_{10}$) =<br>(0.48 +0.48) *($2^2$-1)= 2.9 | FTM ($E_{16}$) =<br>(0.7 + 0.1)*($2^2$-1)= 2.4<br>FTM ($E_{17}$) =<br>(0.1+0.1)*($2^2$-1)= 0.6 |
| FTM ($FT_i$) | FTM ($FT_1$) =<br>(1.11+1.92)*($2^4$-1)= 45.5 | FTM ($FT_2$) = (0.06 + 2.9) * ($2^4$-1) = 44.4 | FTM ($FT_3$) = (2.4+0.6) *($2^4$-1) = 45 |

## 5.2 *Case Study 2: Web Based Contoso University Application*

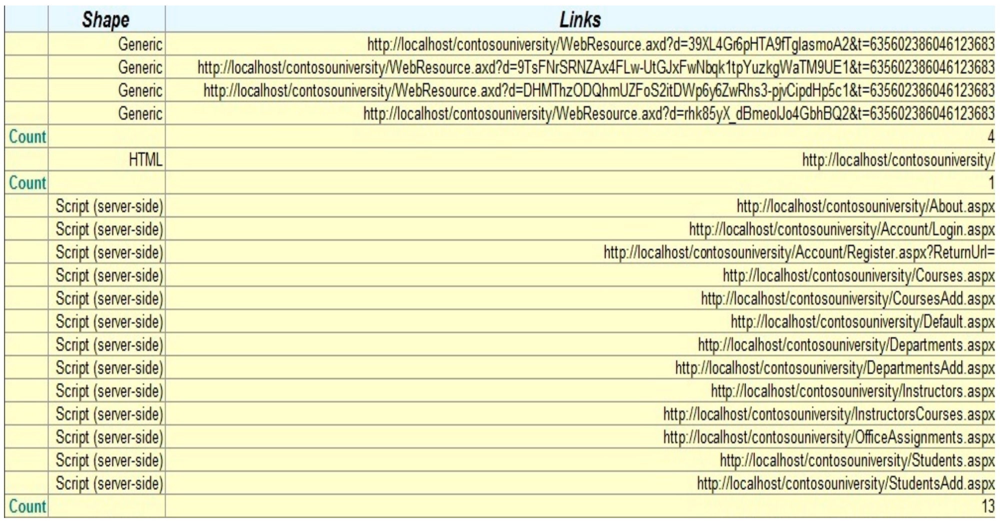
In this section, we present another case study wherein we will compute the FVM of Contoso University application using our proposed approach. We have downloaded the open source code sample web application of the Contoso University from the Microsoft gallery [31]. The project is run on Visual Studio 2010 .NET framework. The web application is developed using ASP.NET. The steps for determining the FVM value of Contoso University application using our approach are as follows:

### 5.2.1. *System Analysis*

There are seven modules in the Contoso University namely Login, Home, About, Students, Courses, Instructors and Departments. The Contoso University application provides access to users of university to view, search and edit information about students, instructors, courses and departments. The login module provides users to access web application. New users can register in Login module. The Home module links to other six modules of the Contoso University application. The About module provides statistics of the student body. The users can view the detailed information about the courses taught at the university in the Courses module. Information about instructors and assigned assignments to them can be viewed in Instructors modules. The user can assign new assignments and remove old assignments assigned to the instructor. Information about departments in the university like name of department, budget, start date, the administrator can be searched and edited by the users in the

Departments module. Figure 15 shows the information about all links in Contoso University parsed by Visio 2003 tool.

The About, Home and Login modules of the Contoso University application have not been identified with any failures. For analysis of web application reliability, we are considering four modules of Contoso University namely Students, Courses, Instructors and Departments. Using a top-down strategy, the application is divided into four modules, i.e. Students, Courses, Instructors and Departments. The statechart diagram for each module is shown in Figure 16, Figure 17, Figure 18 and Figure 19 respectively.



| | Shape | Links |
|---|---|---|
| | Generic | http://localhost/contosouniversity/WebResource.axd?d=39XL4Gr6pHTA9fTglasmoA2&t=635602386046123683 |
| | Generic | http://localhost/contosouniversity/WebResource.axd?d=9TsFNrSRNZAx4FLw-UtGJxFwNbqk1tpYuzkgWaTM9UE1&t=635602386046123683 |
| | Generic | http://localhost/contosouniversity/WebResource.axd?d=DHMThzODQhmUZFoS2itDWp6y6ZwRhs3-pjvCipdHp5c1&t=635602386046123683 |
| | Generic | http://localhost/contosouniversity/WebResource.axd?d=rhk85yX_dBmeolJo4GbhBQ2&t=635602386046123683 |
| Count | | 4 |
| | HTML | http://localhost/contosouniversity/ |
| Count | | 1 |
| | Script (server-side) | http://localhost/contosouniversity/About.aspx |
| | Script (server-side) | http://localhost/contosouniversity/Account/Login.aspx |
| | Script (server-side) | http://localhost/contosouniversity/Account/Register.aspx?ReturnUrl= |
| | Script (server-side) | http://localhost/contosouniversity/Courses.aspx |
| | Script (server-side) | http://localhost/contosouniversity/CoursesAdd.aspx |
| | Script (server-side) | http://localhost/contosouniversity/Default.aspx |
| | Script (server-side) | http://localhost/contosouniversity/Departments.aspx |
| | Script (server-side) | http://localhost/contosouniversity/DepartmentsAdd.aspx |
| | Script (server-side) | http://localhost/contosouniversity/Instructors.aspx |
| | Script (server-side) | http://localhost/contosouniversity/InstructorsCourses.aspx |
| | Script (server-side) | http://localhost/contosouniversity/OfficeAssignments.aspx |
| | Script (server-side) | http://localhost/contosouniversity/Students.aspx |
| | Script (server-side) | http://localhost/contosouniversity/StudentsAdd.aspx |
| Count | | 13 |

Figure 15 Links in Contoso University

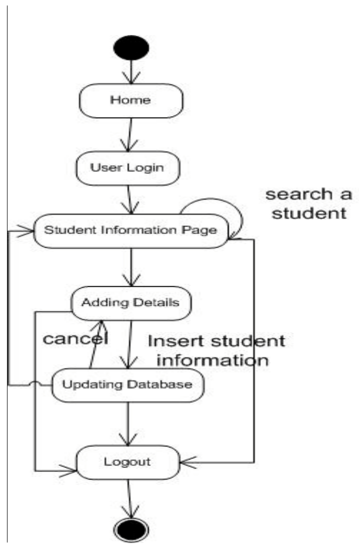


Figure 16 Statechart Diagram for Students

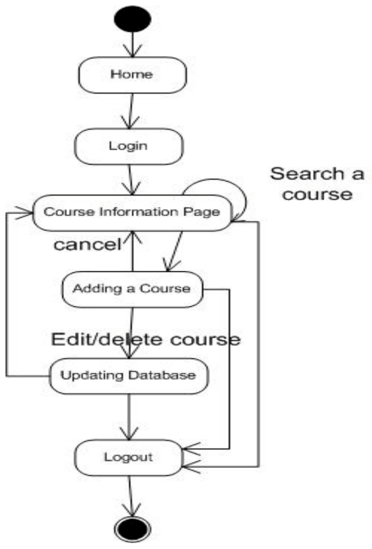


Figure 17 Statechart Diagram for Courses

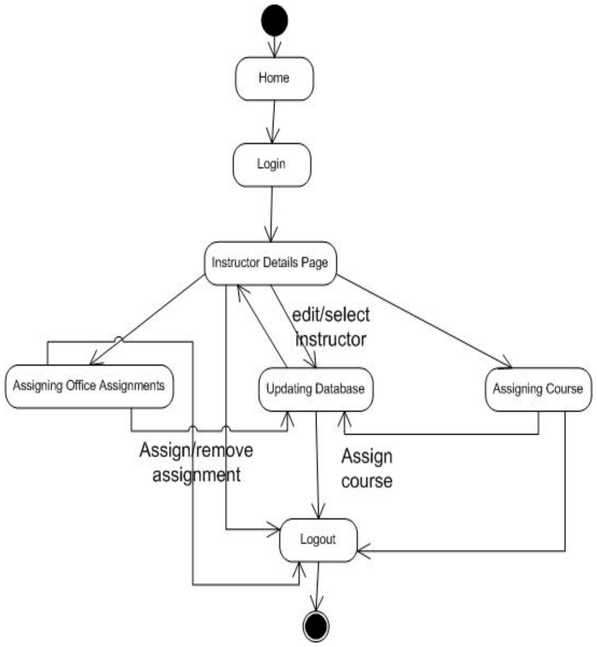


Figure 18 Statechart Diagram for Instructors

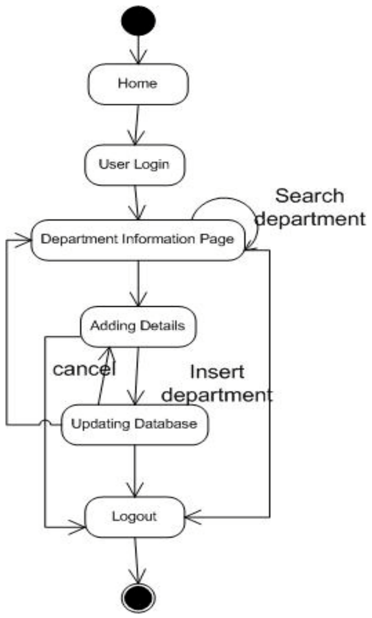


Figure 19 Statechart Diagram for Departments

### *5.2.2. Fault Tree for Contoso University Web Application*

Fault trees are drawn for different modules of the web application. The fault trees are constructed by identifying the fault events and determining the relationship between them for causing the failure event in the particular module.

Table 6 Failures Modes and Mechanism of Modules in Contoso University

| **Module** | **Failure Effect** | **Failure Mode** | **Mechanism** |
|---|---|---|---|
| Students | New student information insertion failed. | • Property field set to null<br>• Invalid data types conversion | Client error, Server Error |
| Courses | New course information insertion failed. | • Property field set to null<br>• Invalid data types conversion<br>• Violation of primary key constraints | Client error, Server Error |
| Instructors | Allotment of office assignments to instructors failed. | • Property field set to null<br>• Invalid data types conversion | Client error, Server Error |
| Departments | Addition of new department failed. | • Property field set to a null value.<br>• Delete statement conflict with reference constraint. | Client error, Server Error |

The system is analyzed and the top events or undesired events in web application are defined. Table 6 shows the faults identified in the Contoso University application. The fault tree constructed for four modules, namely Students, Courses, Instructors and Departments are shown in Figure 20, Figure 21, Figure 22 and Figure 23 respectively.

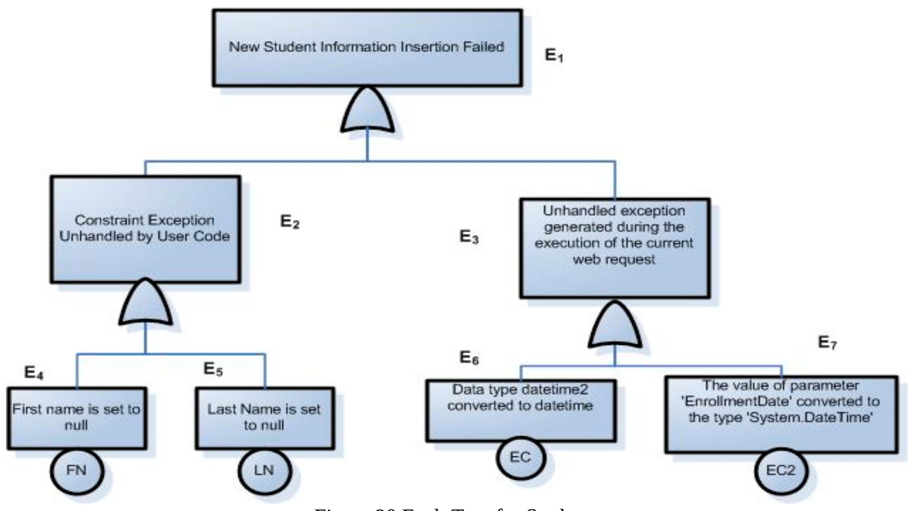

New Student Information Insertion Failed
$E_1$
Constraint Exception Unhandled by User Code
$E_2$
$E_3$
Unhandled exception generated during the execution of the current web request
$E_4$
$E_5$
$E_6$
$E_7$
First name is set to null
Last Name is set to null
Data type datetime2 converted to datetime
The value of parameter 'EnrollmentDate' converted to the type 'System.DateTime'
FN
LN
EC
EC2

Figure 20 Fault Tree for Students

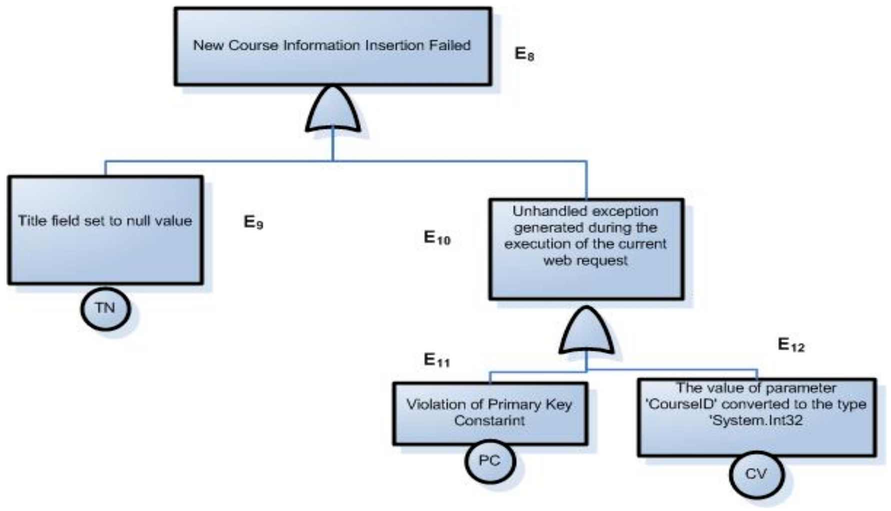

New Course Information Insertion Failed
$E_8$
Title field set to null value
$E_9$
$E_{10}$
Unhandled exception generated during the execution of the current web request
TN
$E_{11}$
$E_{12}$
Violation of Primary Key Constarint
The value of parameter 'CourseID' converted to the type 'System.Int32
PC
CV

Figure 21 Fault Tree for Courses

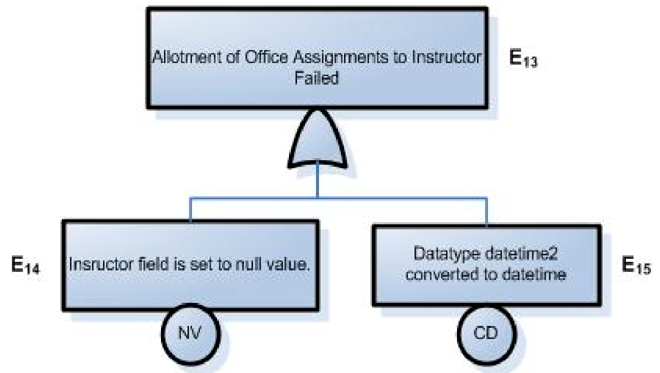

Allotment of Office Assignments to Instructor Failed
E13
Insructor field is set to null value.
E14
NV
Datatype datetime2 converted to datetime
E15
CD

Figure 22 Fault Tree for Instructors

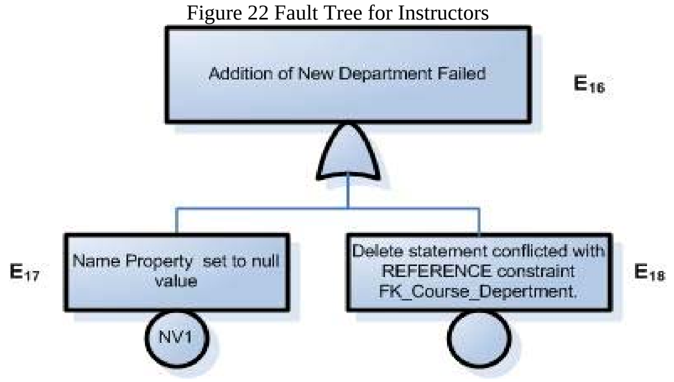

Addition of New Department Failed
E16
Name Property set to null value
E17
NV1
Delete statement conflicted with REFERENCE constraint FK_Course_Depertment.
E18

Figure 23 Fault Tree for Departments

5.2.3. *Determining Reliability of Contoso University Web Application*

Once the fault trees are constructed, next we draw the web navigation graph of the Contoso University application using Microsoft Visio Studio 2003 tool as shown in Figure 24.

5.2.3.1. *Applying Proposed Measures on Fault Tree*

The metrics proposed in section 4 are applied to the fault tree constructed for the Contoso University application. We have applied proposed metrics on the fault tree constructed for four modules, namely Students, Courses, Instructors and Departments as shown in Figure 20, Figure 21, Figure 22 and Figure 23 respectively. The steps for determining FVM of Contoso University application are illustrated in Table 7.

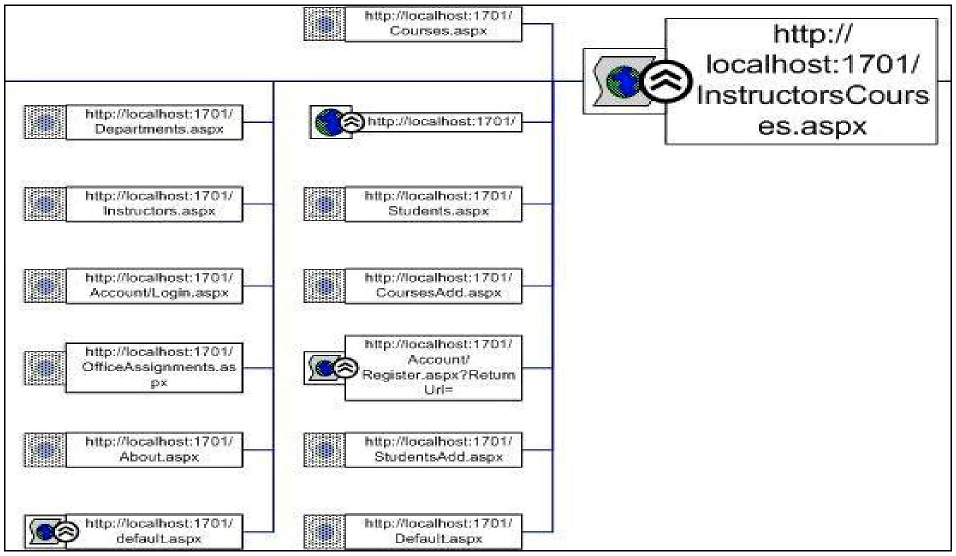

http://localhost:1701/ Courses.aspx
http:// localhost:1701/ InstructorsCourses.aspx
http://localhost:1701/ Departments.aspx
http://localhost:1701/
http://localhost:1701/ Instructors.aspx
http://localhost:1701/ Students.aspx
http://localhost:1701/ Account/Login.aspx
http://localhost:1701/ CoursesAdd.aspx
http://localhost:1701/ OfficeAssignments.aspx
http://localhost:1701/ Account/ Register.aspx?ReturnUrl=
http://localhost:1701/ About.aspx
http://localhost:1701/ StudentsAdd.aspx
http://localhost:1701/ default.aspx
http://localhost:1701/ Default.aspx

Figure 24 Web Page Navigation Graph of Contoso University Web Application

Table 7 Calculation of FTM of different modules of Contoso University Application

| | **MODULE** | | | |
|---|---|---|---|---|
| **Metric** | **Student ($FT_1$)** | **Course ($FT_2$)** | **Instructor ($FT_3$)** | **Department ($FT_4$)** |
| IF ($E_i$) | IF ($E_4$) = 14*14 = 196<br>IF ($E_5$) = 14*14 = 196<br>IF ($E_6$) = 14*14 = 196<br>IF ($E_7$) = 14 *14 = 196 | IF ($E_9$) = 13*14 = 182<br>IF ($E_{11}$) = 13*14 = 182<br>IF ($E_{12}$) = 13*14 = 182 | IF ($E_{14}$) = 13*16 = 208<br>IF ($E_{15}$) = 14*14 = 196 | IF ($E_{17}$) = 14*16 = 224<br>IF ($E_{18}$) = 14*16 = 224 |
| TIF ($FT_i$) | TIF ($FT_1$) = 784 | TIF ($FT_2$) = 546 | TIF ($FT_3$) = 404 | TIF ($FT_4$) = 448 |
| RIF ($E_i$) | RIF ($E_4$) = 196/784 = 0.25<br>RIF ($E_5$) = 196/784 = 0.25<br>RIF ($E_6$) = 196/784 = 0.25<br>RIF ($E_7$) = 196/784 = 0.25<br>RIF($E_4 + E_5 + E_6 + E_7$) = 1.0 | RIF ($E_9$) = 182/546 = 0.33<br>RIF ($E_{11}$) =182/546 = 0.33<br>RIF ($E_{12}$) = 182/546 = 0.33<br>RIF ($E_9 + E_{11} + E_{12}$) = 0.99 | RIF ($E_{14}$) = 208 / 404 = 0.51<br>RIF ($E_{15}$) = 196/ 404 = 0.49<br>RIF($E_{14}$ +$E_{15}$) = 1.0 | RIF ($E_{17}$) = 224/448 = 0.5<br>RIF ($E_{18}$) = 224/448 = 0.5<br>RIF($E_{17}$ +$E_{18}$) = 1.0 |
| FTM($E_i$) | FTM($E_2$) = (0.25+0.25) * ($2^2$1) = 1.5<br><br>FTM ($E_3$) = 1.5 | FTM($E_{10}$) = (0.33+0.33)*($2^2$-1)= 1.98 | ------------- | ------------- |
| FTM ($FT_i$) | FTM ($FT_1$) = (1.5+1.5)*($2^4$-1) = 45 | FTM ($FT_2$) = (0.33+1.98)*($2^3$-1) = 16.17 | FTM ($FT_3$) = (0.51+0.49)*($2^2$-1) = 3 | FTM ($FT_4$) = (0.5+0.5)*($2^2$-1) = 3 |

The RIF of each causative event in a fault tree is determined by applying equation 4. The FTM value of all intermediate events in fault trees corresponding to the modules of the Contoso University application is calculated by applying Algorithm 1 given in Figure 6. The FTM value of each fault tree in Contoso University application is calculated by applying equation 5. Finally the FVM value of the Contoso University system, FVM ($SYS_2$) is determined by applying equation 6 as shown below:

$$\text{FVM}(SYS_2) = 45 + 16.17 + 3 + 3 = 67.17$$

### *5.3 Result Analysis*

According to Aggarwal, K.K. et al. [24], the basis of IF metric is found upon following premises. All but the simplest systems consist of components, and it is the work that these components do and how they are fitted together that influences the complexity of a system [24]. According to system theory, the components that are highly coupled and that lack cohesion tend to be less reliable and less maintainable than those that are loosely coupled and that are cohesive [24]. Coupling is the degree of linkage between one component to others in the same system [24]. According to Jung, W. et al. [4] also complexity of a system is closely related with maintainability, testing efforts, and understandability. In our work, the degree of coupling is defined in terms of number of static links or dynamic links that are input to causative event and a number of static links or dynamic links that output from a causative event in a module. This degree of coupling is then used to measure RIF of each causative event in a fault tree. The value of FTM at event, module and system level is determined from the value of RIF and forms the basis to determine the value of FVM. Based on the concept of system theory, components having a high degree of coupling require more efforts to maintain. In our proposed approach components or module in a web application having high value of FTM or FVM (SYS) therefore is less reliable than the components having low value of FTM or FVM (SYS) and thus require more efforts to maintain.

The FVM values of the Job Portal system and the Contoso University system are 134.9 and 67.17 respectively. The high FVM value of Job Portal System clearly suggests that it is more complex as compared to the Contoso University System. Hence Job Portal system will require more efforts to maintain. Similarly out of four fault trees drawn for the Contoso University application, maximum number of faults is contributed by the fault tree of Student module, i.e., $FT_{1.}$ Therefore, the reliability of $FT_1$ is the lowest which implies that the testing effort or testability of Student module is highest among other modules of the Contoso University application. On the other hand, in the case of Job Portal application, the FTM value of all the three modules is almost equal. This implies that testing effort or maintainability of all the three modules is almost equal.

It can be further concluded that the testability or testing effort of Contoso University application can be reduced if the occurrence of faults due to the Student module can be reduced. For this the designer is advised to redesign the Student module in the Contoso University system. It is to be noted that this information is available before the commencement of actual testing of web application and therefore will help in reducing the testing effort of the web application.

## 6 Conclusion and Future Work

In this paper, an approach to predict the reliability of the web application is proposed using the concept of static link, dynamic link and FTA technique. We have made the first attempt to apply FTA technique for measuring the reliability/testability of web applications. The concept of FTA technique has been used to identify the weak areas or potential failures of web application. In order to do so, the web application is divided into modules where each module is represented by a fault tree. Each fault event in the fault tree leading to undesired event is assigned a measure calculated using a metric named as FTM. The value of FTM is calculated at event, module and system level. The value of FTM at system level is called FVM and can form the basis to predict reliability/testing effort of the web application. However, the areas which need to be redesigned in a module will form a part of the future work. Also in the future we will be extending this approach to compute the value of FTM for any type of event driven systems by combining it with code coverage models like control flow and data flow models. We will extend our approach by analyzing the static source code of web based systems by considering points-to analyses, value elimination, context and control flow analyses. Finally, we also propose to perform theoretical and empirical validation of the proposed metric.

## References

1. Barnhart, C. and Laborte, G. Handbooks in Operations Research & Management Science: Transportation. Elsevier, 2007, 14, First edition.

2. Ghosheh E., Black S. and Qaddour J. Design Metrics for Web Application Maintainability Measurement. in Proceedings of International Conference on Computer Systems and Applications, 2008, 778-784.

3. Helmer, G., Slagell, M., Honavar, V., Miller, L. and Lutz, R. A Software Fault Tree Approach to Requirement Analysis of an Intrusion Detection System. in Symposium on Requirements Engineering for Information Security, 7(4), 2001, 207-220.

4. Jung, W., Lee, E., Kim, K. and Wu, C. A Complexity Metric for Web Applications based on Entropy Theory. in 15th Asia Pacific Software Engineering Conference, IEEE, 2008, 511-518.

5. Leveson, N. G. Software Safety in Embedded Computer Systems. in Communications of the ACM, 34(2), 1991, 34–46.

6. Mao, C., Control and Data Complexity Metrics for Web Service Compositions. in Proceedings of 10th International Conference on Quality Software, Zhangjiajie, IEEE, 2010, 349-352.

7. Marchetto, A. and Trentini, A. Evaluating Web Applications Testability by Combining Metrics and Analogies. in Proceedings of 3rd International Conference on Information and Communication Technology, Cairo, IEEE, 2005, 751-779.

8. Needham, D. and Jones, S. A Software Fault Tree Metric. in 22nd IEEE International Conference on Software Maintenance", IEEE, 2006, 401- 410.

9. Shahzad, A., Raza, S., Azam, M.N., Bilal, K., Inam, U. H. and Shamail, H. Automated Optimum Test Case Generation Using Web Navigation Graphs. in IEEE conference on Emerging Technologies, IEEE, 2009, 427-430.

10. Stamatelatos, M., Vesely, W., Fragola, J., Minarick III, J. and Railsback, J. Fault Tree Handbook with Aerospace Applications. NASA, 2002, version 1.1, available at: http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf, (Last accessed 19 Nov 2014).

11. Thi, Q.P., Quang, D.T. and Quyet, T.H. A Complexity Measure for Web Service. in Proceedings of International Conference on Knowledge and Systems Engineering, Hanoi, IEEE, 2009, 226-231.

12. Tian, P., Wang, J., Zhang, W. and Liu, J. A Fault tree Analysis based Software System Reliability Allocation using Genetic Algorithm Optimization. in World Congress on Software Engineering, IEEE, 2009, 194-198.

13. Towhidnejad, M., Wallace, D.R. and Gallo, A.M. Validation of Object Oriented Software Design with Fault Tree Analysis. in Proceedings of the 28th Annual NASA Goddard Software Engineering Workshop, IEEE, 2003, 209-215.

14. Vesely, W.E., Goldberg, F.F., Roberts, N.H. and Haasl, D. F. Systems and Reliability Research. U. S Nuclear Regulatory Commission, NUREG–0492, 1981, available at: http://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/sr0492.pdf. (Last accessed 20 Nov 2014)

15. Vesely, W.E. A Time Dependent Methodology for Fault Tree Evaluation. in Nuclear Engineering and Design 13, 1970, 337-360.

16. Wang, Y., Teague, T. West, H. and Mannan, S. A New Algorithm for Computer-Aided Fault Tree Synthesis. Journal of Loss Prevention in the Process Industries 15, Elsevier, 2002, 265–277.

17. Xiang, J. and Yanoo, K. Formal Static Fault Tree Analysis. in International Conference on Computer Engineering and Systems, Cairo, IEEE, 2010, 280–286.

18. Ying, R., Hong, L. and Huawei, L. Research on Technique of Software Testing based on Fault Tree Analysis. in International Conference on Computer Science and Network Technology, IEEE, 2011, 1718- 1720.

19. Zhang, Y., Zhu, H. and Greenwood, S. Website Complexity Metrics for Measuring Navigability. in Proc. of the 4th International Conference on Quality Software, IEEE, 2004, 172-279.

20. Li, L., He, T. and Tang, S. Consistency Checking and Test Generation for UML Statechart Diagram via Extended Context-free Grammar. in 6th International Conference on New Trends in Information Science and Service Science and Data Mining, IEEE, 2012, 633-638.

21. Rumbaugh, J., Jacobson, I. and Booch, G. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999.

22. Butkiewicz, M., Madhyastha, H.V. and Sekar, V. Characterizing Web Page Complexity and Its Impact. in IEEE/ACM Transactions on Networking, 22(3), June 2014, 943-956.

23. Panda, S.K., Swain, S.K. and Mall, R. Measuring Web Site Usability Quality Complexity Metrics for Navigability Intelligent Computing, Communication and Devices. in Proceedings of ICCD 2014. Advances in Intelligent Systems and Computing, 308 AISC (Vo1) Springer Verlag, 2015, 393-401.

24. Aggarwal, K.K. and Singh, Y. Software Engineering. New Age International Publishers, 2007, 3rd Edition.

25. Chong, C.W., Ramachandran, V. and Eswaran, C. Web Navigation Efficiency Analysis. in International Conference on Systems, Man, Cybernetics, Vol. 4, IEEE, 1999, 69-73.

26. Leung, K.R. P, Hui, L.C. K., Yiu, S. M. and Tang, R.W.M. Modeling Web Navigation by Statecharts. in COMPSAC, IEEE, 2000, 41-47.

27. Mendes, E., Mosley, N. and Counsell, S. Web Metrics-Estimating Design and Authoring Effort. in Multimedia, 8(1), IEEE, 2001, 50-57.

28. Dhawan, S. and Kumar, R. Analyzing Performance of Web-Based Metrics for Evaluating Reliability and Maintainability of Hypermedia Applications. in International Conference on Broadband Communications, Information Technology and Biomedical Applications, IEEE, 2008, 376-383.

29. Alagappan, B., Alagappan, M. and Danishkumar, S. Web Metrics based on Page Features and Visitor's Web Behavior. in Proceeding of 2nd International Conference on Computer and Electrical Engineering, Vol.2, IEEE, 2009, 236-241.

30. Towhidnejad, M., Wallace, D.R. and Gallo, A.M. Fault Tree Analysis for Software Design. Annual NASA Goddard Software Engineering Workshop, IEEE, 2002, 24-29.

31. https://code.msdn.microsoft.com/aspnet-MVC-application-b01a9fe8.( Last accessed 3 April 2015)

32. Sabharwal, S., Bansal, P. and Aggarwal, M. Modeling the Navigation Behavior of Dynamic Web Applications. International Journal of Computer Applications, 65(13), March 2013, 20-27.

33. Morikawa, I. and Yamaoka, Y. Threat Trees Templates to Ease Difficulties in Threat Modeling. in 14th International Conference on Network based Information Systems, Tirana, IEEE, 2011, 673-678.