



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

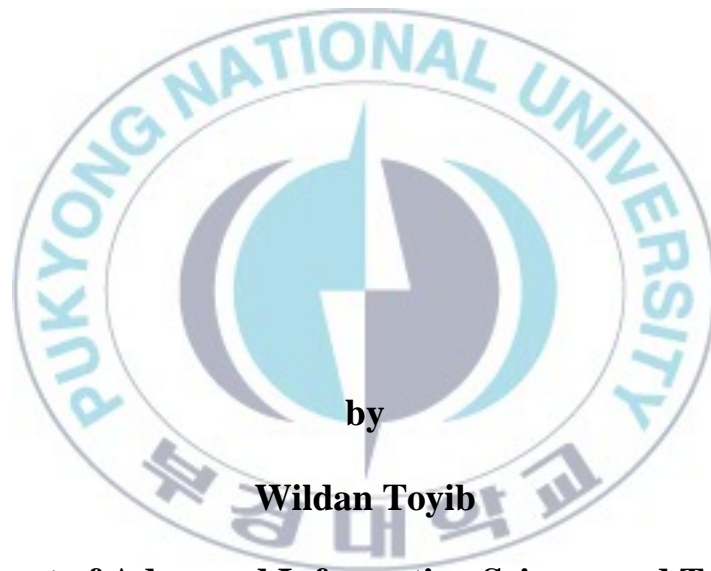
저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Thesis for the Degree of Master of Engineering

**An Integrative Method of FTA and Software
FMEA for Security Analysis of a
Smartphone**



by

Wildan Toyib

Department of Advanced Information Science and Technology

The Graduate School

Pukyong National University

August, 2012

An Integrative Method of FTA and Software FMEA for Security Analysis of a Smartphone

스마트폰의 보안성 분석을 위한 결함 트리 분석과 소프트웨어 고장 모드와 효과 분석의 통합적인 방법

Advisor: Prof. Man-Gon Park

by

Wildan Toyib

**A thesis submitted in partial fulfillment of the requirements
for the degree of**

Master of Engineering

**in the Department of Advanced Information Science and Technology,
The Graduate School,
Pukyong National University**

August, 2012

**An Integrative Method of FTA and Software FMEA for Security
Analysis of a Smartphone**

A dissertation

by

Wildan Toyib

Approved by:



Chang Soo Kim
(Chairman) **Dr. Chang Soo Kim**

Kyung H. Rhee
(Member) **Dr. Kyung Hyune Rhee**

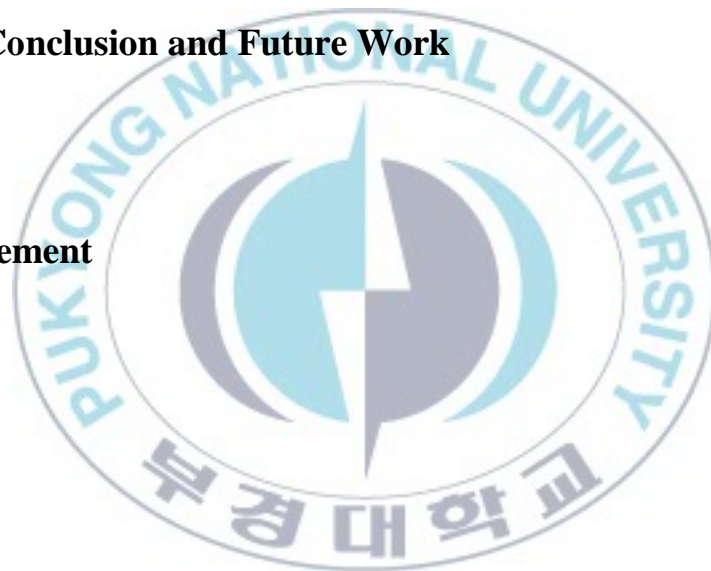
Man-Gon Park
(Member) **Dr. Man-Gon Park**

August 25, 2012

CONTENTS

| | | |
|---|-----------|---------------|
| List of Tables | | iii |
| List of Figures | | iv |
| Abstract | | v |
| Chapter 1. Introduction | | 1 |
| 1.1 Background | | 1 |
| 1.2 Purpose and Structure of the Thesis | | 4 |
| Chapter 2. Smartphone Software Security Mechanisms and Issues | | 7 |
| 2.1 Overview of the smartphone | | 7 |
| 2.2 Business Process Management for Smartphone Security Analysis | | 14 |
| 2.2.1 Contents to be modeled | | 15 |
| 2.2.2 Software Security Analysis Methods | | 16 |
| 2.2.3 Security Analysis | | 17 |
| Chapter 3. Identification of Software Fault, Failure and Error within Smartphone | | 19 |
| 3.1 Identification of Software Faults in Smartphone Environment | | 19 |
| 3.2 Functional Block Diagram for Software Security Analysis | | 21 |
| 3.3 Fault Trees for Mobile Device Security | | 23 |
| Chapter 4. FTA, SFMEA and FMECA for Smartphone Security Analysis | | 27 |
| 4.1 Functions and Methods of FTA | | 27 |
| 4.2 Foundations of SFMEA | | 28 |
| 4.3 Security Analysis regarding to the FTA Cut-Sets | | 31 |
| 4.3.1 Introduction of FTA | | 31 |
| 4.3.2 Cut-Sets | | 34 |
| 4.3.3 Qualitative Assessment | | 35 |
| 4.3.4 Quantitative Assessment | | 35 |
| 4.3.5 Single AND-gate | | 36 |
| 4.3.6 Single OR-gate | | 36 |
| 4.3.7 Cut-sets Assessment | | 37 |
| 4.3.8 TOP Event Probability | | 37 |
| 4.4 Security Analysis related to the SFMEA and FMECA | | 39 |

| | | |
|---|-------|-----------|
| 4.5 FTA and SFMEA Performance and Evaluation in Software Development Lifecycle | | 42 |
| Chapter 5. Integrative Method of FTA and SFMEA for Security Analysis | | 48 |
| 5.1 Integrated forward and backward analysis | | 48 |
| 5.2 Forward Integrated Security Analysis of SFMEA and FTA | | 48 |
| 5.3 Backward Integrated Security Analysis of SFTA and SFMEA | | 51 |
| 5.4 Event Name Populations | | 53 |
| 5.5 Integrative Methods of FTA and SFMEA for Smartphone Security Analysis | | 57 |
| 5.6 FTA, SFMEA and FMECA as Security System | | 72 |
| Chapter 6. Conclusion and Future Work | | 74 |
| References | | 81 |
| Acknowledgement | | 87 |



List of Tables

| | | | |
|------------------|--|-------|----|
| Table 1.1 | Global Sales Figures and Market Share of Smartphone Operating Systems for Third Quarter of 2010 and 2012 | | 2 |
| Table 2.1 | Software Security Mechanism Incorporated in Smartphone Operating System | | 7 |
| Table 3.1 | The Variable Entities of the smartphone fault related to software security system | | 20 |
| Table 3.2 | Event Name of the Mobile Device Security | | 24 |
| Table 4.1 | Cut-set with dependent items | | 34 |
| Table 4.2 | Summary from a real SFMEA report Smartphone | | 43 |
| Table 4.3 | FMEA Worksheet from a real software FMEA report | | 44 |
| Table 5.1 | Table 5.1 : Event Name Failure for Smartphone Security Analysis | | 51 |
| Table 5.2 | Worksheet of the SFMEA Module for Security Analysis on the Smartphone | | 62 |
| Table 5.3 | Data Population for Cut-sets of Risk Priority Number (RPN) within Smartphone Security Analysis | | 65 |
| Table 5.4 | Software Security Variable on Recovery Action | | 66 |
| Table 5.5 | Risk Matrix Analysis for Smartphone's Security Analysis | | 66 |
| Table 5.6 | FTA versus FMECA Selection Criteria | | 67 |
| Table 6.1 | Pseudo Code to develop FTA Cut-Sets by Using Visual Basic Programming Language | | 73 |

List of Figures

| | | |
|--------------------|--|----|
| Figure 1-1 | Business Process Modeling of the Smartphone Security Analysis | 14 |
| Figure 1-2 | Extracting Activities within the Business Process Modeling of the Smart phone Failure Data Analysis in Security Mode | 15 |
| Figure 1-3 | Business Process Modeling for Software Security Analysis of a Smartphone | 16 |
| Figure 3-1 | Functional Block Diagram of Smartphone Security Analysis | 22 |
| Figure 3-2 | Database Configuration for Fault Trees Generator | 24 |
| Figure 3-3 | Fault Trees for Mobile Device Security | 25 |
| Figure 4-1 | Preparations to Deploy for Fault Tree Analysis (FTA) System | 31 |
| Figure 4-2 | Symbols of FTA System | 32 |
| Figure 4-3 | Implementation of Using FMEA in Problem Solving | 38 |
| Figure 5-1 | Forward Integration Security Analysis Technique of Smartphone Failure for SFMEA | 47 |
| Figure 5-2 | Backward integrated security analysis technique of smartphone failure for FMEA | 48 |
| Figure 5-3 | Legend of Sensitivity Value | 52 |
| Figure 5-4 | Fault Trees Analysis (FTA) for Smartphone Failure | 54 |
| Figure 5-5 | Fault Trees of the Smartphone on the Software Failure | 56 |
| Figure 5-6 | Fault Trees of the Smartphone on the Logger Architecture Failure | 57 |
| Figure 5-7 | Fault Trees of the Smartphone on the Hardware Failure | 58 |
| Figure 5-8 | Fault Trees of the Smartphone on the Database Failure | 59 |
| Figure 5-9 | Fault Trees of the Smartphone within Hardware and Software in Severity | 59 |
| Figure 5-10 | Backward Integration of the Smartphone Security Analysis | 64 |
| Figure 6-1 | Developing the transfer symbol shape should have one connector | 80 |

An Integrative Method of FTA and Software FMEA for Security Analysis of a Smartphone

Wildan Toyib

**The Graduate School, Department of Advanced Information Science and
Information Technology
(International Cooperative Program)
Pukyong National University**

Abstract

Recently, software security of the smartphone is an important issue in the field of information science and technology, due to fast propagation of smart technology in our life. Smartphone, as one of the security critical systems which are utilizing for terminal systems of the smart banking, ubiquitous home management, airline passengers screening, map directions, mobile government, disaster detections, are related to the risks of accidents, losses, unavailability, misuses and so on.

The security issues mentioned above, and meanwhile software hazard analysis is the key-approaching concepts for these failures. Fortunately, we propose an integrating efficient architecture for software security analysis of the smartphone by using fault tree analysis (FTA) and software failure mode and effect analysis (SFMEA) to gain a convergence safety and reliability technique on hand handle device. FTA system is interpreted by involving cut-set analysis.

Possibly, there have been a lot of tool and methodology regarding to the software security analysis system methods and concepts. However, for these cases within this research, we consider step taken by business process management to dissect all the fault and failure within an implementation of integrative software of failure mode effect analysis and fault three analysis, this method is the new technique for analyzing and evaluating failure paths in a system, either in a lateral of hardware and or software.

Since the inception of FTA, fault tree theory, methods and computer codes have improved significantly.

The key-concept for this method is to integrate a complex fault of securities inside smartphone in to a fault three cut-sets. To perform a failure mode effect analysis more comprehensive, not only for security reason but also failure problems of this device is a key-method for solution. Thus, in this thesis, we propose an integrative method of FTA and software FMEA for security system of the smartphone by deploying an integrated method is to be a specific software security analysis, this thesis discusses the design and operation feature of FTA and software FMEA, along with its capabilities and benefits.



스마트폰의 보안성 분석을 위한 결함 트리 분석과 소프트웨어 고장 모드와 효과 분석의 통합적인 방법

Wildan Toyib

부경대학교 일반대학원 첨단 정보과학 및 정보기술학과
(국제화협동과정)

요약

최근에 스마트폰의 소프트웨어 보안은 우리의 삶속에서 스마트 기술의 빠른 전파 때문에 정보과학 및 기술 분야에서 아주 중요한 이슈이다. 스마트 은행업무, 유비쿼터스 홈관리, 기내승객 화면업무, 지도안내, 모바일 정부, 재난탐지의 단말기 시스템을 이용하는 중요한 보안 시스템중 하나인 스마트폰은 사고의 위험, 유실, 이용불가능성, 오용등과 관련이 있다. 보안적인 문제들은 위에 것들을 언급하였다. 그 동안에 소프트웨어 위험 분석은 이러한 실패요인들에 있어서 중요하게 다가오는 개념들이다. 다행히도, 우리는 손으로 다루는 기계에서 융합보안, 신뢰성 기술은 얻기 위해서 Fault Tree Analysis(FTA)와 Software Failure Mode and Effect Analysis(SFMEA)를 사용함으로써 스마트폰의 소프트웨어 보안 분석을 위한 효율적인 통합설계를 제안한다. FTA 시스템은 Cut-Set분석을 포함함으로써 해석된다.

가능하게도, 소프트웨어 보안 분석시스템 방법과 개념에 관한 많은 도구와 방

법론이 있었다. 그러나, 이 조사 내 이러한 경우에는 우리는 세 가지 오류 분석, 실패 모드 효과 분석의 통합적인 소프트웨어의 실행에서 모든 오류와 실패를 분해하기 위해서 사업진행관리에 의해 수행된 단계들을 고려한다. 그리고 이 방법은 하드웨어의 측면 혹은 소프트웨어의 측면인 이 시스템 내에서 실패 경로를 평가하거나 분석하는 새로운 기술이다. FTA의 시작으로 인해 fault tree 이론, 방법, 코드들은 상당히 개선되었다.

이 방법의 중요 개념은 스마트폰 내에서 보안의 복잡한 실패요인들을 fault three cut-sets로 통합시키는 것이다. 실패모드 효과분석을 좀 더 이해할 수 있도록 수행하기 위해서, 보안의 이유뿐만 아니라 이 기계의 오류문제 역시 해결책을 위한 중요한 방법이다. 그러므로, 이 이론에서, 통합적인 방법을 배치 함으로써 스마트폰의 보안체제를 위한 소프트웨어 FMEA와 FTA의 통합적인 방법이 특정한 소프트웨어 보안 분석이 된다고 제안하고, 이 이론은 그것의 능력과 이득이 함께하여 FTA, 소프트웨어 FMEA의 설계, 작동 특징을 논의 한다.

Chapter 1

Introduction

1.1 Background

Smartphone is the new generation of mobile and embedded devices, such as mobile phones and PDA (Personal Digital Assistants) which support a rich set of applications, web browsing, SMS-MMS, i-television, i-radio, multimedia and entertainment applications [1]. The time to market pressure forces manufacturers to deliver products with new features within very short time testing (e.g., three months) often sacrificing the testing efforts, as a result. We witness an increasing susceptibility of hand-held devices to accidental errors and malicious attacks. The example is recently reported first mobile phone virus, namely cabir, affecting Symbian. Security becomes even more critical as new critical applications emerge for mobile phones, e.g., robot control [2-3], traffic control [4] telemedicine, pervasive and ubiquitous applications [5]. In such scenarios, a phone failure affecting the application can result in a significant loss or hazard, the robot performing uncontrolled actions on the mobile device remote monitoring system, despite these concerns, very few studies have looked into the dependability of smartphones related to the security system.

There is a few understanding of how and why smartphone fail. This thesis presents software security analysis of a smartphone using integration of FTA, SFMEA (Software Failure Mode Effect Analysis) and FMECA(Failure Mode Effect and Critical Analysis). The analysis starts with a high level event failure characterization of smartphones based on everyday user's experiences. Data for this study spans with two years period

(between 2010 and 2012) and obtained from publicly available web forums, society and communities. Where users post information on their experiences in using smartphone. The information collected in these forums is not well structured yet and relatively small number of entries can be considered as failure reports because of the security mechanism. However, collected data enables consider (1) the characterizations of the failure, occurrence, detection and severity, (2) identification of the high level event failure manifestation, (3) categorization of the user initiated recovery from the device failure, and (4) list of fault related to the security of development tools, web browsers, multimedia and entertainment applications [6].

This initial analysis is then used to guide the development of a failure data logger for smartphones. Initially introduced in the logger employs heartbeat mechanism to detect system and application failures [7]. Upon failure detection, the logger records information about the smartphone activities, presenting the list of fault related to safety and security mechanism because of the human factor, and social networking. Error conditions signal effect by the system or application modules and web browser authentication through hypertext transfer protocol secure (https) within OSI layer [8].

Next analysis is the market leader of smartphone issue products which is depicted in the **Table 1.1**, this analysis is to gain a fault, failure, and error on the smartphone productions and issues.

Table 1.1: Global Sales Figures and Market Share of Smartphone Operating Systems for Third Quarter of 2010 and 2012 [9].

| Platforms | August-November 2010 | | August-November 2011 | |
|--------------|----------------------|--------------|----------------------|--------------|
| | Units/1k | Share[%] | Units/1k | Share[%] |
| Android | 20,544.0↑↑ | 25.3↑↑ | 60,490.4↑↑ | 52.5↑↑ |
| Symbian | 29,480.1↑ | 36.3↓ | 19,500.1↑ | 16.9↑ |
| iOS | 13,484.4↑↑ | 16.6↑↑ | 17,295.3↑ | 15.0↑ |
| RIM | 12,508.3↑↑ | 15.4↑↑ | 12,701.1↑ | 11.0↑ |
| Bada | 920.6↑ | 1.1↑ | 2,478.5↑ | 2.2↑ |
| Microsoft | 2,203.9↑ | 2.7↑ | 1,701.9↓ | 1.5↓ |
| Others | 1,991.3↑ | 2.5↑ | 1,018.1↓ | 0.9↓ |
| Total | 81,132.6 | 100.0 | 115,185.4 | 100.0 |

Legend

↑↑ 2 periods increase

↓ 1 periods decrease

↑ 1 periods increase

↓↓ 2 periods decrease

Based on the **Table 1.1** which represents global market of OS within smartphone, Google and apple are the obvious winners in the smartphone ecosystem. The combined shares of iOS and android in the smartphone Operating System (OS) market double to nearly 62 percent in the second quarter 2010, up from just over 31 percent in the corresponding period of 2011.

In these methods and technologies, FTA which generates the use cases by the minimal cut-sets of fault tree, cannot determine the priorities of all the use cases and cannot utilize the finished software test result. In order to solve these problems, a software security analysis approach with SFMEA which are transferred from fault tree that is described within this thesis.

FTA is an important verification methodology for software security, as a top down technique, FTA can be used to analyze the origin of the failure, determine the software security requirements, defect the software logic errors, identify the multiple failure

sequences involving different parts of the system (hardware, software, and human) and guide the security test. In the software security testing, FTA can be used to determine appropriate input data for testing, and detail the test case definition for the sets of validation test cases to be executed [10].

Some researchers, which were developed, prefer the forward integrated analysis method, i.e. from SFMEA to SFTA [10], meanwhile the others prefer to the backward one, i.e. from SFTA to SFMEA [11]. It is the same with hardware in different integrated directions [12]. Some researchers considered that the forward integrated analysis of hardware is more labor intensive and difficult to apply compared with the backward one. Thus for the software, no final conclusion has yet been reached, and even the principle of the integrated analysis techniques is still under study.

The principle and process both of forward and backward integrated analysis techniques of FTA and SFMEA are being discussed in this thesis; failure of smartphone affected by weak security is the object analysis for this research. However, we propose an integrative method of software FTA and SFMEA for security analysis inside smartphone, this integrative approach is aimed to reduce and classify faults, failures and vulnerable networks. We create a Boolean logic which is applied in the fault tree cut-set symbols, thus find the accurate technique to reduce failure most of them in the smartphone with software FMEA, we assert these methods are very strong related to the security system and technique.

1.2 Purpose and Structure of the Thesis

In this thesis, we create and analysis system by using an integrative method of FTA and SFMEA for security analysis of a smartphone.

Also we create strong analysis regarding to the software security of Smartphone fault, failure and errors by using integrative methods of FTA, SFMEA and FMECA matrix analysis under the consideration (1) forward integrated security analysis of SFMEA and FTA, (2) backward integrated security analysis of SFTA and FMEA, (3) integrative methods of FTA and SFMEA for Smartphone security analysis, (4) FTA, SFMEA and FMECA as security system.

Accordingly, we illustrate the integrative methods of FTA and SFMEA for Smartphone security analysis, by considering to the faults, failures and errors within Smartphone, either in software, hardware and human factors. However, the main focus areas of fault, failure and error in the Smartphones are generated to the Minimal Cut-Sets (MCS) analysis as reliability system.

The structure of this thesis is divided into six chapters as follows:

Chapter 1 introduces background with the related studies and purpose of this thesis.

Chapter 2 introduces basic concept of the software security mechanisms and issues for Smartphone and Business Process Management (BPM) for software security analysis methods.

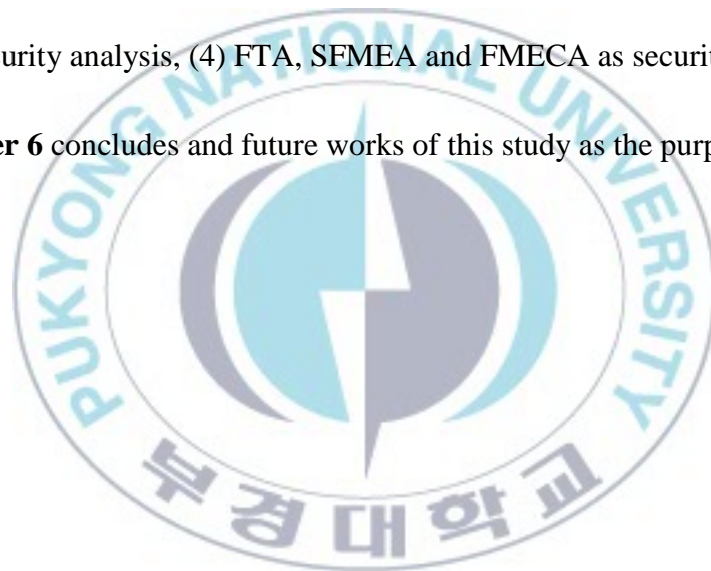
Chapter 3 discusses on identification of software fault, failure and error within Smartphone which consist of identification of software faults in Smartphone circumstance, and Functional Block Diagram (FBD) for Smartphone software security analysis.

Chapter 4 discusses on the introduction of FTA and SFMEA, and FMECA, under

consideration of (1) functions and methods of FTA, (2) foundations of SFMEA, (3) security analysis regarding to the FTA Cut-Sets, (4) security analysis related to the FMEA and FMECA, (5) FTA and FMEA matrix, performance and evaluation system in software lifecycle development process.

Chapter 5 proposes integrative methods of FTA and SFMEA for Smartphone security analysis based on strong Boolean logic approach under consideration of (1) forward integrated security analysis of SFMEA and FTA, (2) backward integrates security analysis of SFTA and FMEA, (3) integrative methods of FTA and SFMEA for Smartphone security analysis, (4) FTA, SFMEA and FMECA as security system.

Finally, **Chapter 6** concludes and future works of this study as the purpose of this thesis.




Chapter 2



Smartphone Software Security Mechanisms and Issues

2.1 Overview of the Smartphone

The top most application layer provides applications such as a phone call, web browser, email client and more applications. Each application in android is packaged in an .apk archive for installation. This archive is similar to a Java standard *.jar files in the way that it holds all code and non-code resources such as images or manifest for the application. Android applications are written in Java based on the API's Software Development Kit (SDK) provides. William Encket al discussed the main components of an android application and how to use an android specific mechanism to protect applications [6]. In general, several security mechanisms are incorporated into the android framework (see **Table 2.1**). We can cluster them into three general groups: mechanism, description, and security issue in detail.

Table 2.1: Software Security Mechanism Incorporated in Smartphone Operating System

| OS | Mechanism | Description | Security Issue |
|---|--|---|---|
|  Linux OS | POSIX users | Each application is associated with a different user ID | Prevents application from disturbing. |
| | File access | Application's directory available to the application. | Prevents application from accessing. |
| | Memory management unit (MMU), Type safety | Each process is running in its own address space. | Prevents privilege escalation, disclosure and denial. |

| | | | |
|---|-----------------------------------|---|--|
|  <p>Android</p> | Type safety | Type safety enforces variable content to adhere to a specific format both in compiling time and runtime. | Prevents buffer overflows and stack smashing. |
| | Mobile carrier security features. | Smartphones use SIM cards to authenticate and authorize user identity. | Prevents phone call theft. |
| | Application permissions | Each application declares which permission it requires at install time. | Limits application abilities to perform malicious behavior |
| | Component encapsulation | Each component in an application (activity or service) has a visibility level that regulates access to it from other applications (i.e., binding to a service). | Prevents one application from disturbing another or accessing private components or API's. |
| | Signing applications | The developer signs application .apk files, and the package manager verifies them. | Verifies that two applications are from the same source. |
|  <p>Symbian OS</p> | Dalvik virtual machine | Each application runs in its own virtual machine. | Prevents buffer overflows, remote code execution and stack smashing. |
| | Beyond client/server sessions | Publish and subscribe also known as properties that provide a mean to define and publish system, and message queues. | Bug on the both of user and kernel side program via similar API's, Message queues offer a peer-to-peer or many-to-many communication paradigm. |
| | Usage scenarios new IPC | Shared buffer I/O drivers no need to have a buffer of their own. | I/O device drivers not need to have a buffer but can share a buffer with a user space process. |
| | Root exploits | Root exploits lib-tif and SMS fuzzing. | Multiple buffer overflows (spams) |
| | Personal data | Aurora feint, mogo road, | The blackmailer, |



iOS

| | | |
|------------------------------|---|---|
| harvesting | Storm8 complaint, Pinch Media | the jealous husband |
| Worms on jail broken devices | This privacy contains of Ikee, dutch 5€ransom, iPhone/Privacy.A, Ikee.B/Duh | Attacks targeting by jail broken iPhones, exploit the fact that very few user bother to change the default root password(alpine) after jail breaking their iPhone and installing a SSH server |

| | | |
|------------------|--|---|
| iPhone forensics | Physical access to any device means that pretty much everything can be compromised with the notable exception of passwords, which encrypted in the phones key chain. | The jealous husband, apple got a lot of bad press, insecure configuration |
|------------------|--|---|



RIM OS

| | | |
|------------|---|--|
| Sandboxing | A virtual container that consists of the memory and the part of the file system that the application process has access to at a specific time | Brute-force attack, online dictionary attack, eavesdropping, impersonating a smartphone, man-in-the middle attack and small subgroup attack. |
|------------|---|--|



Windows Mobile OS

| | | |
|---------------------------|--|--|
| Safeguards | Concerned individuals and organizations aware of the potential risks involved can often mitigate many of the associated threats with add-on security mechanisms. | Loss theft or disposal, because of their small size, handheld devices have a propensity to become lost or misplaced. |
| Maintain physical control | Verifying an individual's claimed identity through user authentication is the first line of defense against unauthorized use of a mobile handheld device. | Unauthorized access, guessing authentication credentials (e.g., a PIN or password. |

| | | |
|--------------------------------|---|---|
| Enable user authentication | Handheld device as the sole repository for important is an invitation for disaster. | Denial of service, key logger, opens https. |
| Backup data | Authentication mechanisms can be bypassed or broken and even deleted information can often be recovered from memory. | Trojan horse, duplication data, failure in encryption process, |
| Reduce data exposure | Malicious programs to mobile phones mainly through communications channels such as multimedia messages or Bluetooth connections, any messages or contacts received on a mobile phone from an unknown number or device should be treated with suspicion. | Malware is typically targeted more toward handheld devices for which a SDK is available than those without one, since code development is easier to perform |
| Shun questionable actions | A simple defense against many forms of malware is to turn off Bluetooth, Wi-Fi, infrared, and other wireless interfaces. | Hijack, tcp-dump, mirror connection, attack on SSID by hacking tools. |
| Curb wireless interfaces | The most direct way of electronic eavesdropping is for spy software to be installed onto a device to collect and forward information within another phone or server | Key-logger, and broken firewall |
| Deactivate compromised devices | Device lost or stolen, disabling service, locking it, or completely erasing its contents is useful actions to take remotely. | Electronic tracking |
| Minimalize functionality | User authentication alternatives including biometric and token-based mechanism. | Cloning into a second cell phone. |
| Add | Memory card | Server-resident |

| | | |
|-----------------------------------|--|---|
| prevention and detection software | encryption, firewall, antivirus, intrusions detection, anti-spam, device content and memory card erasure and virtual private networking. | data is the server was able to be accessed by unauthorized. |
|-----------------------------------|--|---|

According to the **Table 2.1**, a smartphone carrier security features, and telephony systems which have a basic set of attributes and functionalities stemming from a need to identify users, monitor usage and charge the client accordingly, a more general term for these features is AAA (Authentication, Authorization and Accounting), as a smartphone platform, such as an android borrows these classical security features from cellular phone design. Authentication usually occurs via SIM-card and associated protocols. Thus, below we conduct to interpret a software fault for Android, Symbian, iOS, RIM, Windows and so on.

First, android specific security mechanisms, android provides the following dedicated security mechanisms introduced by Google application permissions, component encapsulation and signing, android has roughly 100 built-in API's permissions that control operations ranging from dialing the phone (*CALL_PHONE*), taking pictures (*CAMERA*), using the Internet (*INTERNET*), listening to key strokes (*READ_INPUT_STATE*), and even disabling the phone permanently (*BRICK*) [10-12].

At installation, the system grants permissions that the installed application requests based on checks of that application's signature against those of the applications declaring the permissions. After the user has installed the application and it receives permissions, it cannot longer request any more permission. Devices have become more and more the target of hacker's malicious software due to growing capabilities of

smartphone, there are certain risks of device which can be called risk of costs, risk of loss, risk of availability, and risk by usage [10][13-14].

Second, Symbian security features like public key signatures on applications and others root CA's in ROM. It also others different spaces for the kernel and the user space and in particular Symbian support access controls such as SIM-PIN, a device security code and Bluetooth are pairing with a key. So how does Symbian handle the risks? 'Risk of costs', this is hardly handled by Symbian. Symbian users should be smart enough not to accept unsigned apps. 'Risk of Loss', Symbian has no functionality to remotely shut-down a lost or stolen device. 'Risk of Availability', this risk is hardly covered as any application can render the phone unusable which has been proven by the scull - 12 -rojan [10]. The last risk of Symbian is viruses, well known scull - 12 -rojan, it comes camouflaged as an extended theme manager, has to be installed by the user. When installed, every link on the mobile phone will be replaced by a scull making the device unstable. Another virus is the proof of concept virus EPOC, well known cabir, which came out in 2004, and spreads over Bluetooth, since then, hundreds of viruses have come into being [11-12].

Thirde is Apple iOS, within this OS every application runs in its own sandbox and terminates when the user presses the 'Home' button. iOS runs a daemon called the 'security server' which implements several security protocols like access to key chain items. With the 'App Store' Apple provides the only possibility of loading programs on to your device. These programs are checked by Apple before they appear in the 'App Store' so iOS users can be quite sure malware will not be found there. But a lot of people have jail broken their iOS is allowing to load software on to their device without going through the 'App Store'; this indeed is risky as you may install instable software.

Let us have a look at how iOS handles the risks. ‘Risk of cost’, since Apple controls which app lands in the app store you can say that you are quite secure not to load a dialer or something like that on to your iPhone. ‘Risk of losses, iOS handles this risk if you are connected to a Microsoft exchange server. Then you can remotely wipe your phone from the server side, iOS showed this feature as a highlight on their SDK. ‘Risk of availability’, iOS covers this very well too. First, the applications run inside their own sandbox without contact to other parts of the iOS than the permitted space. Second, iOS checks every application goes to the ‘App Store’ so harmful software will not find the way on to the iOS [13][15-17].

Finally, Windows mobile edition, there are studies that show the windows mobile edition seems to be just as vulnerable as Symbian, if not even more [12]. Microsoft began as well to implement security policies with signed applications, but unfortunately these policies do not work as advertised because of bugs, consequently vulnerability still exists. In addition to that, the application unlock for windows mobile edition needs no knowledge and takes at least ten minutes. A lot of windows mobile devices may be unlocked as open source software is mostly unsigned [12][16][18-19]. How does a window mobile cover our risks? ‘Risk of costs’, users either install the malicious software itself or they may spread through Wi-Fi or Bluetooth. A dialer could use this for doing extensive harm [12]. ‘Risk of loss’: Windows mobile has no possibility of being shut down remotely. ‘Risk of availability’: Windows mobile is vulnerable here as a virus may access the data on the device. Windows mobile users can take a deep breath now as in the beginning of 2008 only 5 viruses existed [13][19].

2.2 Business Process Management for Smartphone Security Analysis

A smartphone is a smart device with complexity system; these figures will describe business process inside of the software security of smartphone. In this process, step taken is starting with business process modeling, then each node will be verified to extract functions, most of these processes are main idea for the research subject. On the **Figure 1-1** is business process management of the smartphone failure, process re-engineering is being embarked from in service by integrating of FTA and software FMEA, a collection related to, structure activities or tasks that produce a specific service or product for a particular object [20-21]. In this case, the object is a failure event in the smartphone which is affected by fault because of vulnerable security analysis.

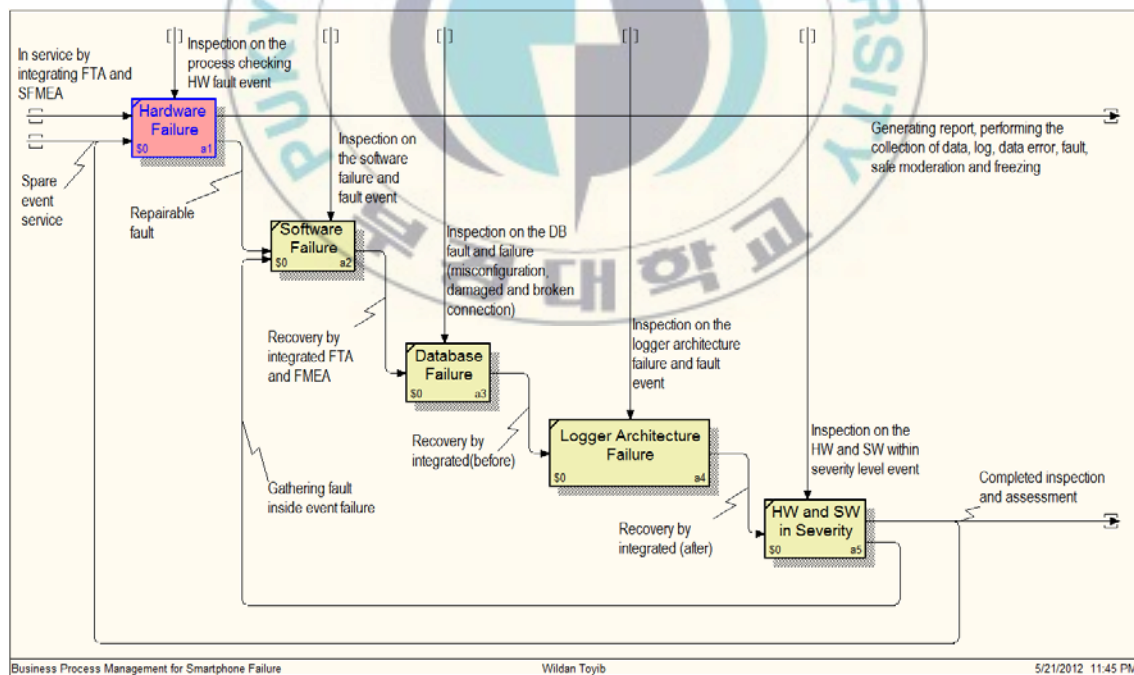


Figure 1-1: Business Process Modeling of the Smartphone Security Analysis

However, to explain all the business process modeling related to the smartphone

security analysis, we design an organization existing resources as depicted on the **Figure 1-1**, this drive for realizing dramatic improvements by fundamentally re-thinking how the security analysis work should be done distinguishes re-engineering from process improvement efforts that focus on functional or incremental improvement [22-23], based on this figure above, we would like to impress the best solutions through flows approaching in reducing failure inside of the smartphone features, both of the analysis, neither depicted on the **Figure 1-1** and the business process products in particular, we have already five aspects for analyzing system, the five aspects are such as hardware failure [a1], software failure [a2], database failure [a3], logger architecture failure [a4] and hardware and software in severity [a5], fifth of them are chain of the development resource of failure within this thesis.

Finally, the goals both of the figures are classifying in the software and hardware system on the issues of the smartphone security analysis. However, we also implement the basic concepts of the business process are such as mechanism, controlling, input and output objective of the failure event which already discovery in the figures above.

2.2.1 Contents to be modeled

Through our methodologies, the following work content information is modeled. This information is strictly defined through the meta-model and notation we use, as explained below such as.

a) Business structure

The layered structure and relations between businesses are described. A “business structure diagram” based on the Unified Modeling Language (UML) class diagram (package diagram) is used to describe the structures. Each business is expressed as a

UML package in this diagram. Information regarding to the following business processes, business data, etc. is described in detail.

b) Business processes and process flows

Each of the pieces of work (tasks) which constitute a business is put in order according to the work dependency to create what is called a “business process”. A flow of business processes is then described (a “business process flow”). The candidate for description includes not only the work that a system does, but also the work which people carry out.

2.2.2 Software Security Analysis Methods

The software failure modes and effects analysis has function to analyze software components or interactions between software and hardware components.

| Software Faults | Failures |
|----------------------------|---------------------------|
| - Data sampling rate | - Broken sensors |
| - Data collisions | - Memory overwritten |
| - Illegal commands | - Missing parameters |
| - Commands out of sequence | - Parameters out of range |
| - Time delays, deadlines | - Bad input |
| - Multiple events | - Power fluctuations |
| - Safe modes | - Gamma radiation |

The Fault Tree is a graphic model of the pathways within a system that can lead to a foreseeable, undesirable loss event; the pathways interconnect contributory events and conditions, using standard logic symbols. Numerical probabilities of occurrence can be entered and propagated through the model to evaluate probability of the foreseeable,

undesirable event, only one of many system safety analytical tools and techniques. Fault tree analysis is best applied to cases with large, perceived threats of loss, i.e. high risk, numerous potential of contributors to a mishap. Complex or multi element systems or processes, already identified undesirable events and indiscernible mishap causes, i.e. autopsies.

In the 1950s, the military and aerospace industries started to develop and use predictive safety analysis techniques; Identify hazards, eliminate and reduce, or control hazardous conditions, to avoid or lessen the severity of accidents.

2.2.3 Security Analysis

Different security analysis techniques address different aspects of the problem.

- Identify hazards
- Demonstrate the absence of specific hazards.
- Determine the possible damaging effects resulting from hazards
- Determine the causes of a hazard
- Identify safety design criteria that will eliminate, reduce, or control identified hazards.
- Evaluate the adequacy of hazard controls

Most security-analysis techniques are aimed at hardware failures or at external threats such as:

| Failure Modes and Effects Analysis (FMEA) | Hazard Analyses |
|--|--|
| <ul style="list-style-type: none"> - Identify critical hardware components, interfaces - Identify possible failure modes for each critical component - Determine the worst-case effect from each failure mode | <ul style="list-style-type: none"> - Identify environmental hazards - Identify deviations in designs - Identify potential deviations in operational use - Identify failures at interfaces between components |

Software does not break, Software failures are due to logic or design errors:

- The software has no coding errors, but is written from incorrect requirements
- The requirements are correct, but the software has coding errors that deviate from requirements.

Thus, software-security processes often try to improve software security by improving software correctness.

Chapter 3

Identification of Software Fault, Failure and Error within Smartphone

3.1. Identification of Software Faults in Smartphone Environment

The identification of software faults for security analysis is based on the Fault Tree Analysis (FTA) technique [5]. Our process is comprised of four main steps. In the first step the Software Requirements Fault Tree (SRFT) is generated which identifies the security faults in the software requirements. This fault tree is then verified, validated and corrected if necessary in the second steps. In the third step security requirements are generated and the original software requirements specification is modified to comply with the security requirements. In the fourth step the security of the resulting software requirements is verified and validated. These steps are iterative and they can be performed any number of times during the process [14]. We provide entities for Smartphone failure regarding to the security system. The explanation of the list of fault is an analysis in hardware both of software system as a basic system for application, the entities is divided in to two categories, hardware and software. That's why we can collect many fault caused of panic neither failure activities that make easy to vulnerable a security within software system on the Smartphone log activities. Based on the **Table 3.1** will describe all the activities, what does it means a smartphone getting failure due of a security? The core function below is an introduction which can run on hardware and software, these failures are considered for security effect, and however, we will

integrate by using a FTA and Software FMEA.

Table 3.1 The Variable Entities of the Smartphone fault related to software security system

| Panic/Failure | Fault Meaning |
|----------------------------|---|
| KERN-EXEC | This Panic is raised when the kernel execute cannot find an object in the object index. |
| | This panic is raised when an unhandled exception occurs, and causes. |
| | This Panic is raised when a timer event is requested from an asynchronous timer service, an Rtimer , and a timer event is already outstanding. It is caused by calling either the At () , After () or Lock () members. |
| E32USER-Cbase | This panic is raised by the destructor of a Cobject . |
| | This panic is raised by an active scheduler, a CactiveScheduler . |
| | This panic is raised by the Error () , CactiveScheduler , RunL() , Error() and CactiveScheduler . |
| | This panic is raised if no trap handler has been installed as CtrapCleanup::New () |
| USER | This panic may be raised by the Left () , Right () , Mid () , Insert () , Delete () , and Replace () . |
| | It may be caused by any of the copying, appending or formatting member functions and, specifically, by the Insert() , Replace() , Fill() , Fillz() , ZeroTerminate() , and SetLength() function |
| KERN-SRV | This panic is raised by the kernel server when it attempts to close a kernel object in response to a RhandleBase::Close () request. |
| ViewSrv | Occurs when one active object event handler monopolizes the thread active scheduler loop and the applications ViewSrv active cannot respond in time. |
| EIKON-LISTBOX | Occur when using a list box object from the eikon framework and no view is defined to display the object |
| | Occur when using a list box object from the eikon framework and an invalid current item index specified |
| Virus Attacker | Such as an intensive IP address scan/sweep attack on MS can evoke a paging storm, and consequently a connection setup storm, which would overload the mobile network equipment such the Radio Network Controller (RNC) and Serving GPRS Support Node (SGSN). |
| Fast Dormancy | Such as RRC state such as IDLE , CELL_PCH , URA_PCH , and CELL_FACH . |
| Phone.app | Bugs on algorithm. |
| EIKOCTL | Corrupt Edwin state for in lining editing. |
| MSGs Client | Failed to write data into asynchronous call descriptor to be passed back to client |
| Always On line PDP Context | Always on line application requires a permanent IP connection, such as Deactivation Accept, Deactivation Ignore and Re-Activate PDP After Deactivation. |
| Always On line Application | Smartphone allows people to access the internet anytime anywhere for any kind of service, for real-time web services, pull/polling, long polling and Push. |

3.2. Functional Block Diagram for Software Security Analysis

The Functional Block Diagram (FBD) describes a function between input and output variables. Function is described as a set of elementary block, Input and output variables are connected to blocks by connection lines. In this method we classify the main category within control panel, type of operating system, hardware and software system, data logger, five key aspects of security, threat model of security, three class of target, and threat attack model for smartphone device security [19]. The performance illustrates on the **Figure 3-1** where FDB describes the security method for handle in security analysis which is extracted with four block diagram, on the other words, the first block will perform hardware system detail which is caused the smartphone failure, second block is containing of data logger which have function as a parameter for detection methods, the third is software system which is the critical security for this issue in kernel management server, and the fourth is a block for network infrastructure with the correlation of information security area, aspect security, threat, target and model of attack, for detail performance, it will illustrate as below completely with the accessory user access, cloud internet infrastructure, implementation of power management, applying panic detector engine, meanwhile the operating system will be getting safe mode. These behaviors appear as a mark that system in recovery mode, for further detail information the flow is depicted on the **Figure 3-1** below.

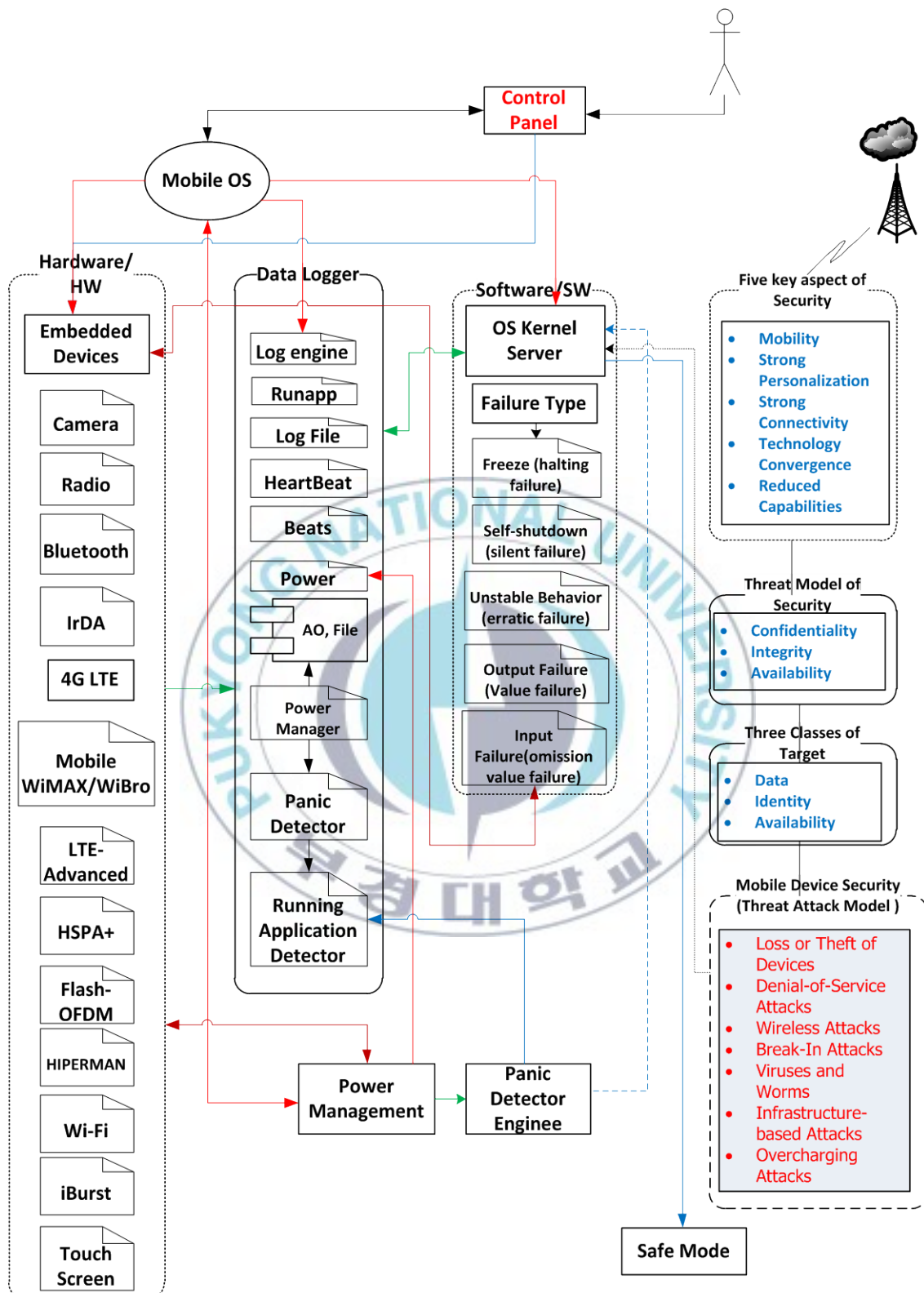


Figure 3-1: Functional Block Diagram of Smartphone Security Analysis

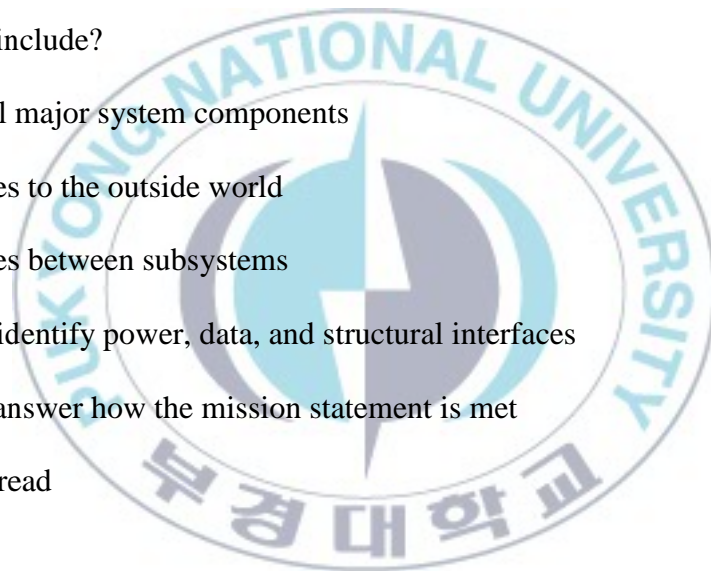
On the other word, not only about the hardware, software and data logger functional

diagram is an explanation for this block function, but also, we perform key aspect of security such as mobility hand handle device, strong personalization within network, strong connectivity with another tools as Bluetooth, IrDA, and RFID, then technology convergence which means a central information provided for anywhere, anytime with smart device accessing information, finally the key concept of security is a capabilities to reduce an attack both of virus neither worm.

A function block diagram describes a function between inputs and outputs and should describe the system in one picture as depicted on the **Figure 3-1** above.

What should it include?

- Show all major system components
- Interfaces to the outside world
- Interfaces between subsystems
- Clearly identify power, data, and structural interfaces
- Should answer how the mission statement is met
- Easy to read



3.3. Fault Trees for Mobile Device Security

Accordingly, to the functional block diagram which is depicted on the Figure 3-1, we can consider by using the data analysis for the for mobile device security features. However, to raise the fault tree system in to inductive security analysis we did, we have to populate data in to data ware house before generating by ODBC (Open Database Connectivity) in to fault tree cut-sets analysis. Thus, we can consider main point related to the mobile device security such as:

- Threat attack models (Lost or theft of device and DoS attack)

- Attack groups as are, wi-fi attack, break in attack, viruses and worms, overcharging attack, and infrastructure based attacks.

For this reason, we generate the **Table 3-2** accordingly the numerous logic analyses is implemented to discovery event name, within the mobile device security such as depicted on the **Figure 3-1**. Further, data generator which is filling on the **Table 3-2** as shown below.

Table 3.2: Event Name of the Mobile Device Security

| No. | Event Name | Q mean | Failures | FVImp | RDF | RIF | FC | Sensitivity Value |
|-----|------------------------------|----------|----------|----------|----------|---------|-----------|-------------------|
| 1 | Threat attack model | 0.000315 | | | | | | |
| 1.1 | Lost or theft of device | 0.000485 | 0.059406 | 0.000423 | 1.084584 | 458.764 | 0.000541 | 9.87880 |
| 1.2 | DoS attack | 0.000634 | 0.000027 | 0.000565 | 1.099046 | 457.304 | 0.0340349 | 7.90094 |
| 1.3 | WiFi attack | 0.000043 | 0.000456 | 0.000953 | 1.009305 | 456.099 | 0.0945945 | 7.09035 |
| 1.4 | Break-In attack | 0.000089 | 0.000445 | 0.000343 | 1.035904 | 455.999 | 0.009099 | 1.09094 |
| 1.5 | Viruses and Worms | 0.000987 | 0.000566 | 0.000545 | 1.009094 | 467.099 | 0.0908989 | 7.00900 |
| 1.6 | Infrastructure based attacks | 0.000898 | 0.000465 | 0.000344 | 1.045805 | 367.098 | 0.084545 | 9.09090 |
| 1.7 | Overcharging attacks | 0.000456 | 0.000456 | 0.000534 | 1.040905 | 456.098 | 0.0680943 | 5.00900 |

Note:

- Code equal to event name
- Occurrence equal to number of occurrences of the basic event in all minimal cut sets.
- FV importance equal to Fussell-Vesely Importance (FV=Q of MCS which contains the basic event/Q of all MCS)
- FC equal to fractional contribution of basic event (1-1/RDF)
- RDF equal to risk decrease factor
- RIF equal to risk increase factor
- Sensitivity equal to sensitivity value, calculated with sensitivity factor = 10

Sensitivity analysis allowing the automatic variation of event failure and repair data between specified limits [24].

With numerous logic is applied to discovery fault tree analysis system, such as Q mean (quality mean), after unavailability calculation and MCS analysis, importance and

sensitivity analysis may be performed. Importance analysis results help to select those fault tree events, which contribute most to the system's unavailability. Sensitivity analysis helps to choose those events, where a relatively small change will lead to a relatively large system unavailability changes. Calculated values are Fussell-Vesely importance (FV Imp), Risk Decrease Factor (RDF), Fractional Contribution (FC), Risk Increase Factor (RIF) and Sensitivity Value for each Basic or Undeveloped Event [25], to generate data source DB by involving between two software systems, we can consider data source deployment by using ODBC as depicted in the **Figure 3.2** below.

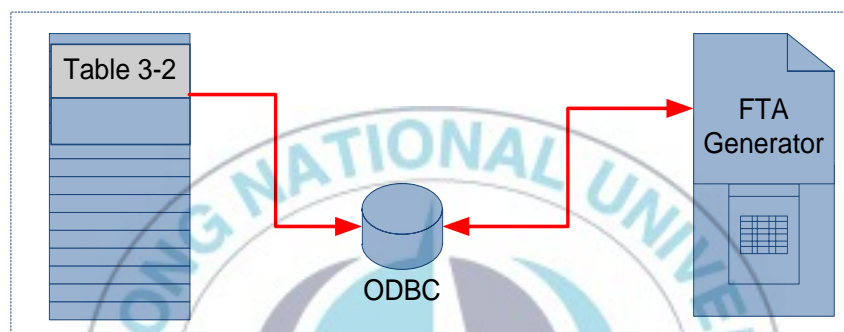


Figure 3-2: Database Configuration for Fault Trees Generator

To generate reports which have been populated in to the DB mobile device failure, we have to plot the main point of failure, preparing a framework for designing. Then, we build fault tree using VB language by configured to the Microsoft Visio. In particular, now we can generate the FTA cut-sets in to the framework that already designed as depicted on the **Figure 3-3** below.

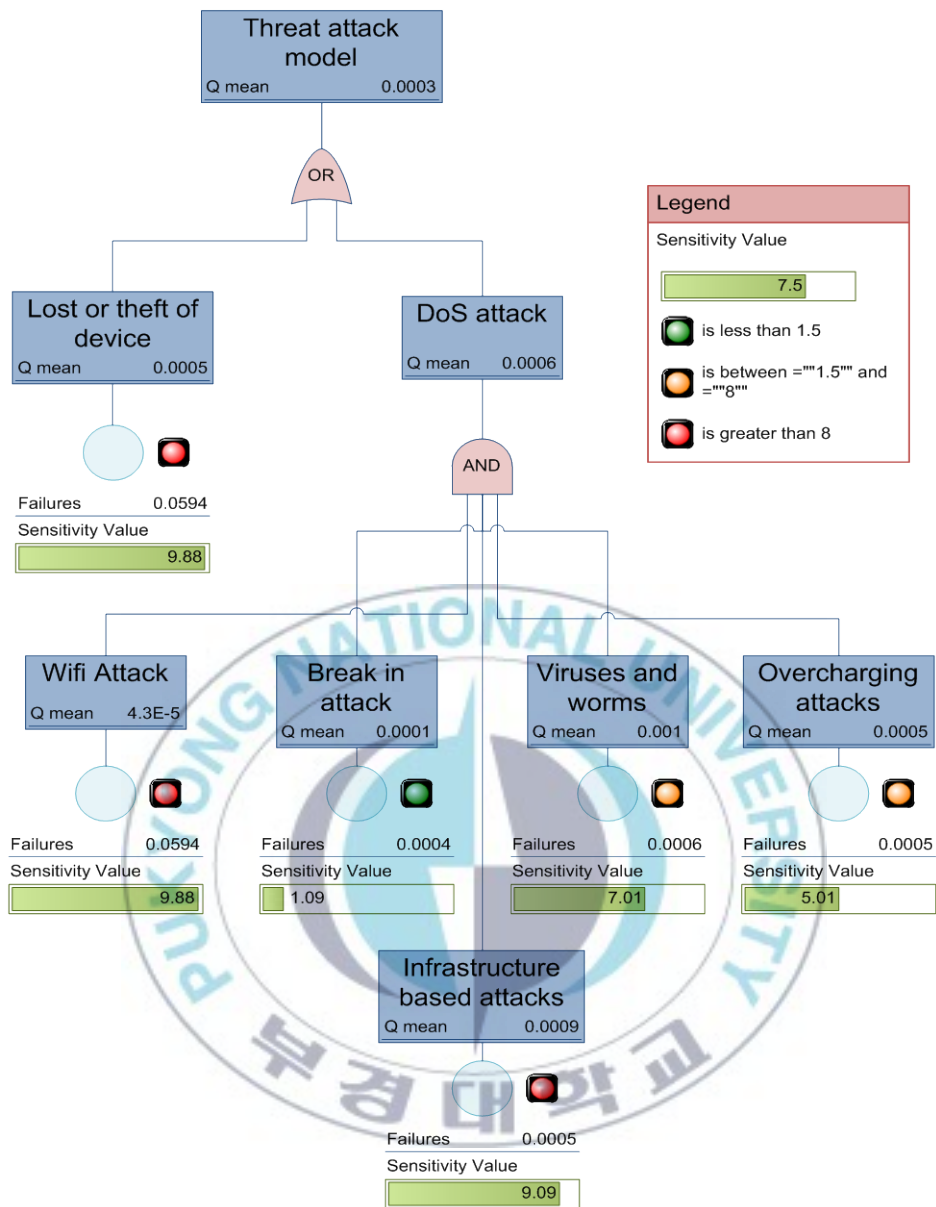


Figure 3-3: Fault Trees for Mobile Device Security

Chapter 4

FTA, SFMEA and FMECA for Smartphone Security Analysis

4.1. Functions and Methods of FTA

The FTA method is widely used to analysis the source of dangers in aerospace, electrical engineering, and nuclear industries. The events on the labels of the structure must be predicted and verified by other technologies, FTA uses Boolean logic to explain the combination of individual errors that can be lead to dangerous event. Each level of the three is needed to show the causes of the problems addressed in the upper level, and it categorizes many basic events [26].

FTA is a deductive, top-down method aimed at analyzing the effects of initiating faults and events on a complex system. This contrasts with failure mode and effects analysis (FMEA), which is an inductive, bottom-up analysis method aimed at analyzing the effects of single component or function failures on equipment or subsystems. FTA is very good at showing how resistant a system is to single or multiple initiating faults. It is not good at finding all possible initiating faults. FMEA is good at exhaustively cataloging initiating faults, and identifying their local effects. It is not good at examining multiple failures or their effects at a system level. FTA considers external events; FMEA does not [27]. In civil aerospace the usual practice is to perform both FTA and FMEA, with a Failure Mode Effects Summary (FMES) as the interface between FMEA and FTA.

However, we should understand regarding to FTA module features so well, such as:

- Up-to-date, intuitive and powerful fault tree diagram interface allowing full control over the diagram: elements location, colors, styles, zooms, etc.
- Handy methods for diagram printing and simple copy & paste transfer to other applications
- Easy to use events Library
- Generation of minimal cut sets
- Calculation of unavailability $Q(t)$, mean unavailability Q
- Calculation of importance and sensitivity
- Calculation of frequency $W(t)$ and intensity $L(t)$
- Calculation of unreliability $F(t)$ and number of failures $E(0,t)$
- Set of required reports - FTA diagram, MCS, events library etc.
- Link between FTA and the product tree
- Link between FTA and FMECA modules
- Integration with security analysis module
- Automatically build FTA from FMECA
- Automatically build FTA from FMEA
- Data import from Risk Spectrum, Aralia SimTree and CAFTA and MS Excel [22-23][28].

4.2. Foundations of SFMEA

The SFMEA process was found to be successful in identifying some ambiguous, inconsistent and missing requirements. More importantly, the SFMEA process, followed by a backward analysis somewhat similar to Fault Tree Analysis (FTA),

identified four significant, unresolved requirements issues. These issues involved complex system interfaces and unanticipated dependencies. Our results challenge some current views on the limitations of SFMEA and suggest that recent efforts by researchers to integrate SFMEA with a broader FTA approach have merit [27][31-32].

Typically, FMEA is practiced on physical systems, and the failure modes considered are the failures of physical components, caused by wear or other damage to the system. Since software has been introduced into automotive and hand-handle device systems, it has often been included in FMEA reports as a component with no failure modes (the ECU containing the software), and the system design FMEA has been produced assuming that the software works correctly.

The concept of software failure mode and effects analysis (software FMEA) has grown in attractiveness over recent years as a way of assessing the reliability of software. Like its hardware counterpart, software FMEA is immensely tedious for an engineer to perform, as well as being error-prone. Clearly, software components do not fail in the same manner as hardware components — a function or method does not break over time because it has become worn or damaged. Software FMEA considers all potential faults such as faulty inputs or software bugs (mutations) that could exist and ensures the worst-case consequences are known, possibly prompting actions to reduce risk. A software bug may be treated analogously to a hardware component failure, the essential difference being that hardware failures occur over time whereas software bugs exist undetected but usually only affects a very small (untested) region of the overall system behavior.

One of the unique difficulties with software systems is the complex relationship between faults and effects. A minor fault can, for example, cause a complete crash of a

software system or have almost invisible but very complex, subtle, and long lasting side effects. The result is that software often has very non-uniform quality in terms of the effects of potential failures, and it is not clear when effort is available, where it should be expended to improve quality. An FMEA provides just this information allowing targeting of effort at the highest risk areas.

Code-level software FMEA has been performed for some years [1, 2, 3, 4], but has been considered impractical except when applied to small pieces of highly critical code, because of its cost. On the other hand, software FMEA of a more abstract specification of the system can ignore important implications of failures, especially where code is not automatically generated from the abstract specification.

Functional interpretation [29] is a vital part of organizing and abstracting the results of structural and behavioral analysis into the form of an FMEA report able to flag significant potential problems. Information regarding the purpose(s) of the system is required, and a functional model is used to provide this information in a principled form. The functional model described in this section identifies the purposes of the system by means of associations with the system interface.

The function interpretation language allows functions to be decomposed into subsidiary functions to build a functional hierarchy. A good deal of intuition regarding the function model can be obtained from an example functional description for a simple smartphone error detection log program shown below.

-
- (1) **FUNCTION** error_detections
 - (2) ACHIEVES set_logic_function
 - (3) BY run_program
 - (4) TRIGGERS display_list_errors
 - (5) AND print_report
-

-
- (6) FUNCTION create_detection
 - (7) ACHIEVES allow_reverse_errors
 - (8) BY run_program
 - (9) TRIGGERS address_detection
-
- (10) FUNCTION display_list_errors
 - (11) ACHIEVES verify_reverse_errors
 - (12) BY show_list_errors_on_screen
 - (13) AND
 - (14) show_bugs_on_screen
-
- (15) FUNCTION print_detection
 - (16) ACHIEVES segregate_fault_error_failure_into_folders
 - (17) BY print_in_figures
 - (18) AND
 - (19) print_in_text
 - (20) AND
 - (21) print_details
-

4.3. Security Analysis related to the FTA Cut-Sets

4.3.1 Introduction of FTA

Fault Tree Analysis (FTA) is a top-down approach to failure analysis, starting with a potential undesirable event (accident) called a TOP event, and then determining all the ways it can happen. The analysis proceeds by determining how the TOP event can be caused by individual or combined lower level failures or events. The causes of the TOP event are “connected” through logic gates, in this thesis we only consider AND-gates and OR-gates, FTA are the most commonly used technique for causal analysis in risk and reliability studies.

FTA was first used by Bell Telephone Laboratories in connection with the security analysis of the Minuteman missile launch control system in 1962, Technique improved by Boeing Company, Extensively used and extended during the reactor security study (WASH 1400) [30].

FTA has main step for analysis such as definition of the system, the TOP event (the

potential accident), and the boundary conditions, construction of the fault tree, identification of the minimal cut sets, qualitative analysis of the fault tree, quantitative analysis of the fault tree, and reporting of results [31].

We have many way to express fault, failure and error that why to prepare for FTA, we have to consider the following statement, the starting point of an FTA is often an existing FMECA and a system block diagram, the FMECA is an essential first step in understanding the system, the design, operation, and environment of the system must be evaluated and the cause and effect relationships leading to the TOP event must be identified and understood.

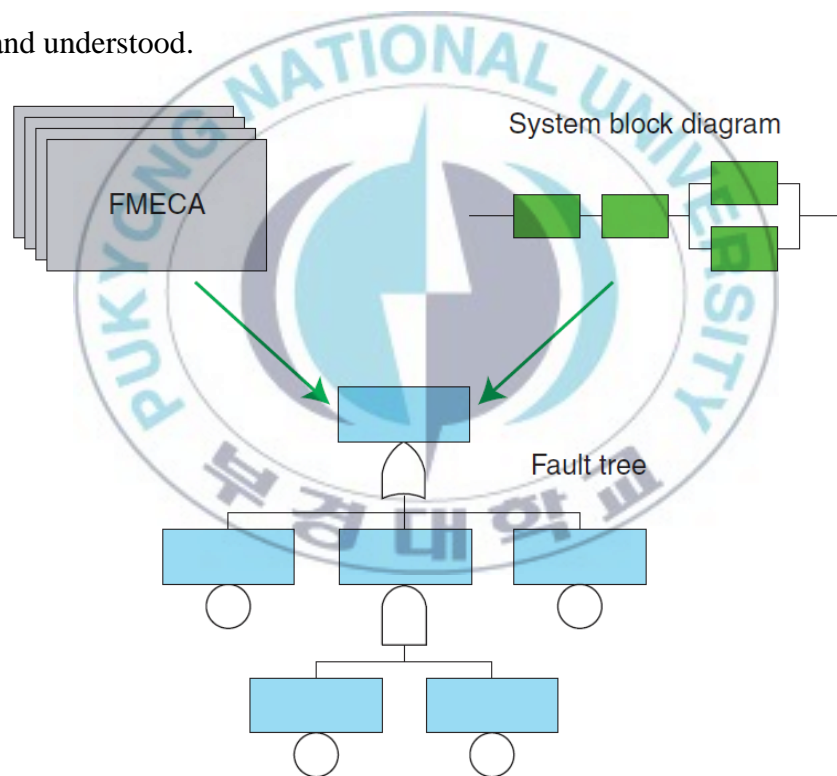


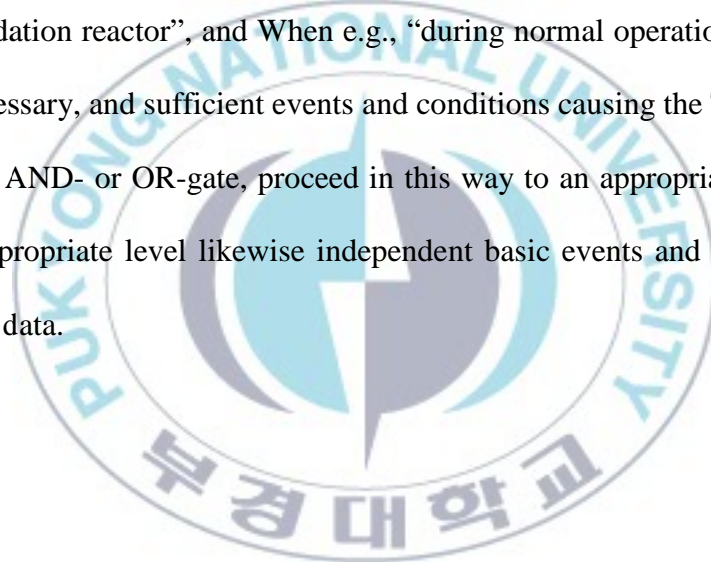
Figure 4-1: Preparations to Deploy for Fault Tree Analysis (FTA) System

Regarding to the **Figure 4-1** above, we can consider the boundary conditions to build FTA in preparing configuration system, actually we have to observe the physical boundaries of the system (Which parts of the system are included in the analysis, and

which parts are not?). The initial conditions (What is the operational stat of the system when the TOP event is occurring ?), Boundary conditions with respect to external stresses (What type of external stresses should be included in the analysis-war, sabotage, earthquake, lightning, etc.?), and The level of resolution (How detailed should the analysis be ?).

In software engineering system, we have to understand very well relate to construction design; however FTA also has fault tree construction such as define the TOP event in a clear and unambiguous way, should always answer: What e.g., “Fire”, Where e.g., “in the process oxidation reactor”, and When e.g., “during normal operation”. What are the immediate, necessary, and sufficient events and conditions causing the TOP event?

Connecting via AND- or OR-gate, proceed in this way to an appropriate level (= basic events), and appropriate level likewise independent basic events and events for which we have failure data.




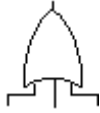
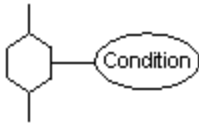




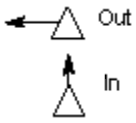
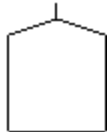
| Symbol | Name | Meaning |
|---|-------------------------|--|
|  | And gate | Event above happens only if all events below happen. |
|  | Or gate | Event above happens if one or more of events below are met. |
|  | Inhibit gate | Event above happens if event below happens and conditions described in oval happen. |
|  | Combination gate | Event that results from combination of events passing through gate below it. |
|  | Basic event | Event that does not have any contributory events. |
|  | Undeveloped basic event | Event that does have contributory events, but which are not shown. |
|  | Remote basic event | Event that does have contributory events, but which are shown in another diagram. |
|  | Transferred event | A link to another diagram or to another part of the same diagram. |
|  | Switch | Used to include or exclude other parts of the diagram which may or may not apply in specific situations. |

Figure 4-2: Symbols of FTA System

4.3.2 Cut-Sets

A cut set in a fault tree is a set of basic events whose (simultaneous) occurrence ensures that the TOP event occurs, a cut set is said to be minimal if the set cannot be reduced

without losing its status as a cut set. The TOP event will therefore occur if all the basic events in a minimal cut set occur at the same time.

4.3.3 Qualitative assessment

Qualitative assessment by investigation the minimal cut-sets :

- Order of the cut-sets
- Ranking based on the type of basic events involved
 - Human error (most critical)
 - Failure of active equipment
 - Failure of passive equipment
- Then, we can also look for large cut-sets with dependent items

Table 4.1 : Cut-sets with dependent items

| Ranking | Basic event 1 | Basic Event 2 |
|----------------|-------------------------|-------------------------|
| 1 | Human error | Human error |
| 2 | Human error | Failure of active unit |
| 3 | Human error | Failure of passive unit |
| 4 | Failure of active unit | Failure of active unit |
| 5 | Failure of active unit | Failure of passive unit |
| 6 | Failure of passive unit | Failure of passive unit |

- Performed by means of Minimal Cut Sets (MCS) building

4.3.4 Quantitative Assessment

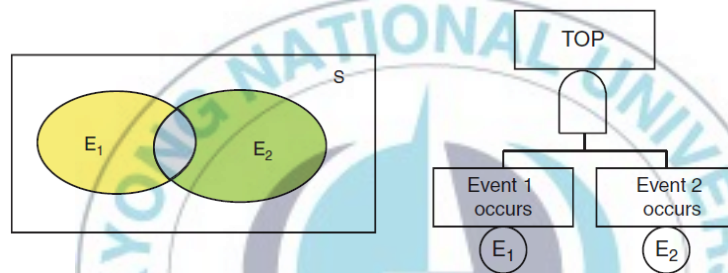
$$Q_0(t) = \Pr (\text{the TOP events occurs at time } t)$$

$$Q_i(t) = \Pr (\text{Basic event } i \text{ occurs at time } t)$$

$$Q_j(t) = \Pr (\text{Minimal cut set } j \text{ fails at time } t)$$

Let's $E_i(t)$ denote that basic event i occurs at time t . $E_i(t)$ may, for example, be that component i is in a failed state at time t . Note that $E_i(t)$ does not mean that component i fails exactly at time t , but that component i is in a failed state at time t , a minimal cut set is said to fail when all the basic events occur are present at the same time. On the other hand, quantitative analysis is to calculating the absolute probabilities, i.e. the probabilities of system failures [32-33].

4.3.5 Single AND-gate



Let $E_i(t)$ denote that event E_i occurs at time t , and let $q_i(t) = \Pr(E_i(t))$ for $i = 1, 2$. When the basic events are independent, the TOP event probability $Q_0(t)$ is

$$Q_0(t) = \Pr(E_1(t) \cap (E_2(t))) = \Pr(E_1(t)) \cdot \Pr(E_2(t)) = q_1(t) \cdot q_2(t) \dots \dots \dots (4.1)$$

When we have a single AND-gate with m basic events, we get

$$Q_0(t) = \prod_{j=1}^m q_j(t) \dots \dots \dots (4.2)$$

4.3.6 Single OR-gate

When the basic events are independent, the TOP event probability $Q_0(t)$ is

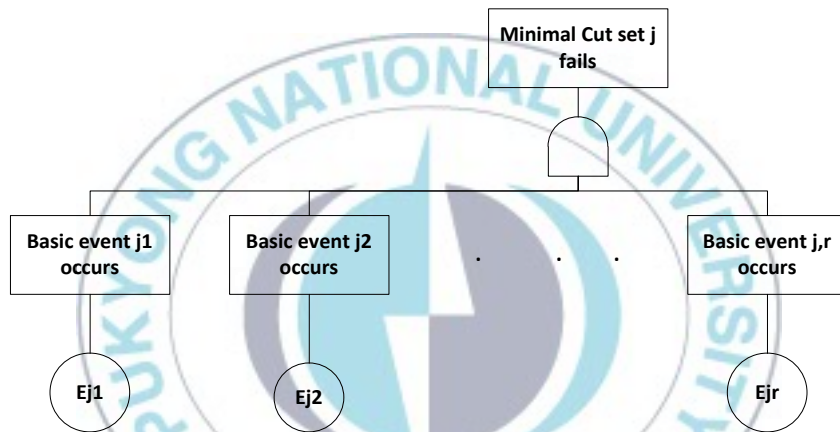
$$Q_0(t) = \Pr(E_1(t) \cup E_2(t)) = \Pr(E_1(t)) + \Pr(E_2(t)) - \Pr(E_1(t) \cap E_2(t)) \dots \dots \dots (4.3)$$

$$= q_1(t) + q_2(t) - q_1(t) \cdot q_2(t) = 1 - (1 - q_1(t))(1 - q_2(t))$$

When we have a single OR-gate with m basic events, we get

$$Q_0(t) = 1 - \prod_{j=1}^m (1 - q_j(t)) \dots \dots \dots (4.4)$$

4.3.7 Cut Set Assessment



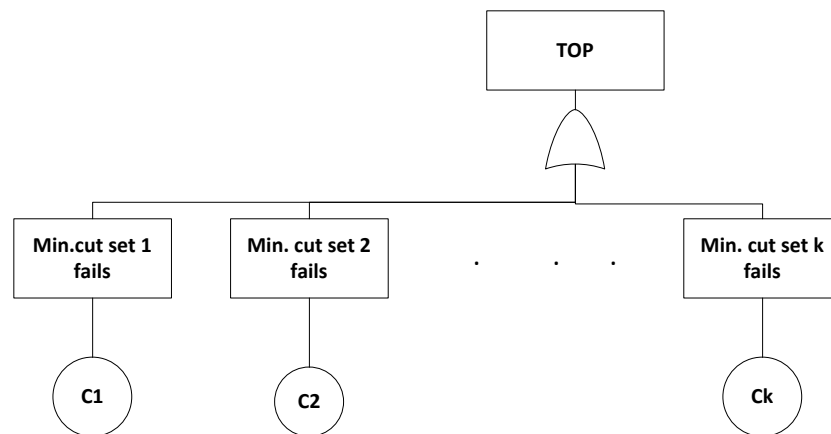
A minimal cut set fails if and only if all the basic events in the set fail at the same time.

The probability that cut set j fails at time t is

$$Q_j(t) = \prod_{i=1}^r q_{j,i}(t) \dots \dots \dots (4.5)$$

Where we assume that all the r basic events in the minimal cut set j are independent.

4.3.8 TOP Event Probability



The TOP event occurs if at least one of the minimal cut sets fails, the TOP event probability is

$$Q_0(t) \leq 1 - \prod_{j=1}^k (1 - Q_j(t)) \dots \dots \dots (4.6)$$

The reason for the inequality sign is that the minimal cut sets are not always independent. The same basic event may be member of several cut sets. Formula (1) is called the upper bound approximation.

Finally, we can evaluate Cut set identification based on ranking of minimal cut sets:

- Cut set unavailability, the probability that a specific cut set is in a failed state at time t
- Cut set importance, the conditional probability that cut set is failed at time t , given that the system is failed at time t .

However, we have already known regarding to the FTA and cut set extraction process, hence the conclusion of the FTA is:

- FTA identifies all the possible causes of a specified undesired event (TOP event)
- FTA is a structured top-down deductive analysis.
- FTA leads to improved understanding of system characteristics. Design flaws and insufficient operational and maintenance procedures may be revealed and

corrected during the fault tree construction.

- FTA is not (fully) suitable for modeling dynamic scenarios.
- FTA is binary (fail–success) and may therefore fail to address some problems.

4.4. Security Analysis related to the SFMEA and FMECA

Failure Mode Effects Analysis, or FMEA as it is commonly called, is a simple method for finding out the real cost of potential failures in any product or system. FMEA can be used during design or later analysis of a product or process to help identify potentially significant failure risks. For example, an engine casing may be found to be at risk of cracking under harsh vibration or an order entry system may lose customer details if the wrong computer key is pressed. It is a scalable tool that can be used to examine failures in complete systems, subsystems or on individual components. The level and depth of analysis should depend on what is being examined and on the importance of finding all key risks. FMEA is used to identify and prioritize how items fail, and the effects of failure.

- When to use it: Situations when it is useful.
- How to understand it: Details of how it works.
- Examples: Some examples of usage.
- How to do it: Step-by-step instructions on using it.
- Practical variations: Variants and variation.

Hence, we will explain regard to the main point above, how can we understand about FMEA in real life and essentials items.

When we would like to use it, to use it when designing products or processes, to

identify and avoid failure-prone designs. Use it when investigating why existing systems have failed, to help identify possible causes and remedies. Use it when investigating possible solutions, to help select one with an acceptable risk for the known benefit of implementing it. And Use it when planning actions, in order to identify risks in the plan and hence identify countermeasures as depicted in the **Figure 4-3** below.

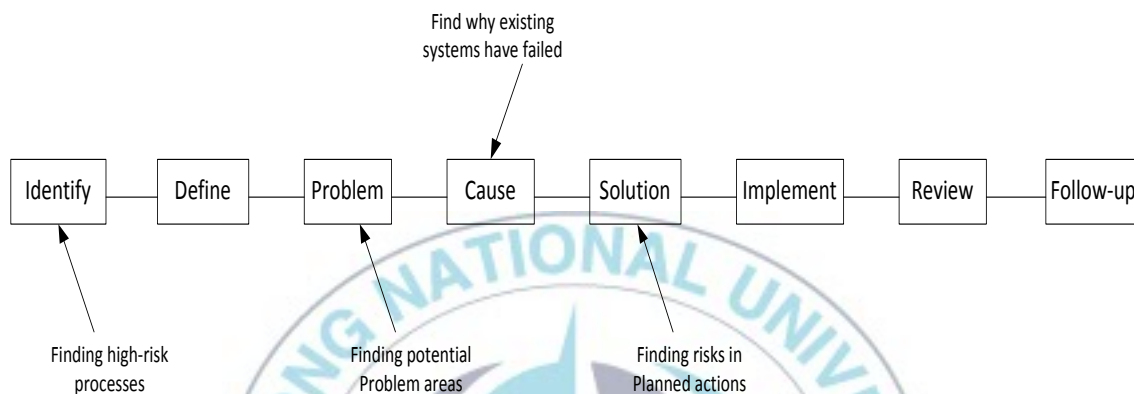


Figure 4-3: Implementation of Using SFMEA in Problem Solving

How we would like to understand it, many problems are caused by systems which fail in unexpected ways, which can result in significant costs. An example of this could be where a new roofing compound is decomposed by acid rain, with the result that the manufacturers have to pay substantial warranty costs, as well as gaining a reputation for poor products. Detailed analysis of the possible way in which a system might fail, and the possible effects of these failures, may thus save significant future costs. Failure mode and effects analysis (commonly called FMEA) takes the dual step of first finding out how an item can fail, and then finding what effect this failure might have.

Examples, a developer of a word processor package received a number of complaints from its customer base about some specific features. On further investigation, it found that there were a limited number of effects that particularly annoyed customers which

working with their customers, they allocated severity ratings.

How we would like to do it, select the item to be analyzed. If it is a part of another item, then be clear about the boundary. For example, if the item is 'vehicle doors', it may mean passenger doors, but not the tailgate, Identify the overall approach to be used. The FMEA may be a part of a larger set of failure analyses. In this case, the way that items are selected needs to be determined, typical strategies include:

- Top-down analysis, where the system being analyzed is broken into pieces and FMEAs done on the larger items first.
- Bottom-up analysis, where the analyses of the smallest pieces are done first, followed by the higher level assemblies from which these are made.
- Component analysis, where the FMEAs are done on the physical parts of the system.
- Functional analysis, where the analysis is of the intended functions and operation of the system.

Identify the scope of failure to be examined, design an appropriate table to capture the right information, identify items which may fail and which fall into the scope defined, if doing criticality analysis, determine the chance of failure for each items and if doing criticality analysis, identify the proportion of the time during the scope described.

Finally, practical variations which have several areas such as focus first on a limited set of failure effects, and then work back to find the modes that cause them, so these can be addressed. Identify a set of severity ratings for failure effects and show the criticality of each item in a separate column. Use a Matrix Diagram to correlate failure modes and failure effects (put items and modes in rows, put effects in columns). Actually, MIL-

STD-1629 describes two methods of FMEA. ‘Method 101’ covers basic qualitative FMEA, whilst ‘Method 102’ covers the quantitative criticality calculation [34-37].

4.5. FTA and SFMEA Performance and Evaluation in Software Development Lifecycle

Why Perform Software FMEA? While most developers reasonably assume that “software doesn’t fail,” we all know that things sometimes do go wrong as a processor executes its code – a memory location can be unintentionally overwritten, algorithmic errors and timing problems can occur, processor or interface circuits can fail, and bad data can be received from the outside world. This is why software-related catastrophic failures sometimes make headlines, even though the failed software had been subject to highly stringent security requirements for development and test [38].

To help prevent such catastrophes, the security analysis looks for worst-case system effects when any one software failure occurs, and in particular to determine whether a single failure can result in a catastrophic event. Since you cannot test for every potential failure, the FMEA takes software quality beyond what qualification testing achieves. It helps avoid the far greater expense of fixing problems after system delivery, potentially ruinous expenses of catastrophic failures – and headlines.

SFMEA is intrinsically tedious and potentially confusing, but a structured approach and specially tailored database tools make the process feasible, highly accurate, and very thorough.

What we provide;

- System-level description documents

- Analysis requirements
- System design requirements
- Source code, if analysis is code level
- Design capture documentation (UML, text, other)
- Design description documents
- Developer feedback in response to analyst questions

The steps at the right summarize the analysis process. We can consider with available materials to support the analysis at the level we desire – code level, method level, or class level.

- We apply our experience to perform steps 1, 3, 4, and 5.
- We use our FMEA database toolset for steps 2 and 6.

During the FMEA process, our analysts will advise your developers of any suspected software flaws or weaknesses as we discover them.

FMEA process, step by step

- 1) Become Familiar with System and Software
 - Use tools and established guidelines
 - Identify where more information is needed
 - Obtain needed info, make necessary assumptions
- 2) Capture Data in Database Tools
 - Customize database for analysis level and analysis requirements
 - Develop shorthand annotation where applicable
- 3) Develop Rules and Assumptions
 - Build upon experience
 - Involve entire analysis team
- 4) Develop Descriptive Failure Modes
 - Determine ways that elements can fail
 - Build a table to avoid duplicate descriptions

5) Determine How Individual Failures Affect System

- Examine elements subject to failure, one by one
- Select appropriate system fail modes from menus using hardware failure modes where possible

6) Generate the Report

- Generate tables (real samples are shown here)
- Summarize FMEA ground rules and assumptions
- Write additional report material
- Assemble the report, save on electronic media

Finally, generate software FMEA report, this is part of a summary table from a real software FMEA report, it lists system failures and identifies software failures that cause them.

Table 4.2: Summary from a real SFMEA report

| System Failure Effect | Qty | Software Failures Causing Possible System Failures FMEA Table IDs | Qty | Software Failures Causing Definite System Failures FMEA Table IDs |
|---|-----|---|-----|---|
| “Hard-over” fin (CRITICAL FAILURE) | 47 | 7.09, 7.10, 7.11, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.32, 7.33, 7.34, 7.35, 7.36, 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.53, 7.55, 7.57, 7.60, 7.61, 7.63, 7.68, 7.69, 24.61 | 0 | |
| SRM inadvertently armed (CRITICAL FAILURE) | 1 | 12.39 | 0 | |
| WH inadvertently activated (CRITICAL FAILURE) | 12 | 7.56, 7.58, 7.64, 12.14, 12.64, 15.54, 15.56, 15.58, 15.61, 15.62, 15.65, 20.11 | 0 | |
| Archived data incorrect or incomplete | 14 | 7.54, 7.59, 7.62, 7.65, 7.66, 7.67, 16.29, 16.31, 16.32, 16.33, 19.18, 19.21, 19.31, 24.45 | 97 | 15.09, 15.10, 15.11, 15.12, 15.13, 15.14, 15.15, 15.16, 15.17, 15.18, 15.19, 15.21, 15.22, 15.23, 15.24, 15.25, 15.26, 15.27, 15.28, 15.29, 15.30, 15.31, 15.32, 15.33, 15.34, 15.35, 15.36, 15.37, 15.38, 15.39, 15.40, 15.42, 16.02, 16.03, 16.04, 16.05, 16.06, 16.07, 16.08, 16.10, 16.11, 16.13, |

This is part of a FMEA worksheet from a real software FMEA report; the key items listed include failure elements (variables), failure modes and system effects.

How Software FMEA complements structured developments. Software failure modes and effects analysis considers what DO-178B guidelines [39-40] and other structured developments do not: potential software failures without regard to requirements, algorithms, timing, or anything else that might not show up even with the most thorough robustness testing. The FMEA's focus is on potential problems that cannot be found during testing because no testing program can force each thing that could go wrong to actually go wrong, and no testing program can set each variable to all values under all system states.

Table 4.3: FMEA Worksheet from a real SFMEA report

| ID | 12.13 | 12.14 | 12.15 | 12.16 | 12.17 |
|----------------------|--|--|--|---|--|
| Input variable | DeltaTimeScale Factor | InbufCmdNumber | InbufCmdNumber | CommandProcSubsTblAdr | InbufCmdNumber |
| Type | float | int | int | addr | int |
| HW | None | None | None | None | None |
| Function | Used as a constant to Scale time differences. 0.1/ 76.8. | Command sent by IGU. (Position of bit set in Inbuf command bytes). | Command sent by IGU. (Position of bit set in Inbuf command bytes). | Table for indexed call (Used as Constant). | Command sent by IGU. (Position of bit set in Inbuf command bytes). |
| Failure Mode | Incorrect Value | Value equal to WHACTIVAT ECMDNO | Value not equal to WHACTIVAT ECMDNO | Incorrect Address | Incorrect Address |
| Failure Cause | Hardware / Software Failure | Hardware / Software Failure | Hardware / Software Failure | Hardware / Software Failure | Hardware / Software Failure |
| Local Failure Effect | The delta between LastFinTim and last fintime will be incorrect. | #WHACounters will not be zeroed when they should be. | Counters will be zeroed. | Program will jump to a random location in memory. May cause | If not the command it should be then the wrong handling |

| | | | | | |
|------------------------------|---|--|----------------|---|---|
| | | | | major problems. | subroutine will be called. |
| System Failure Effect | Definite incorrect time or sequence data returned to IGU. | Possible WH Inadvertently activated. Definite loss of WH counters interlock. | Unpredictable. | Unpredictable. | Unpredictable. |
| Module | CommandProc | CommandProc | CommandProc | CommandProc | CommandProc |
| Line | 270 | 284 | 284 | 297 | 297 |
| Notes/Suggestions | None. | None. | None. | This should be checked. The affects on variables cannot be predicted. | May end up in a random place in code. This value should be checked. Used as offset into CommandProcSub. |

In software FMEA at the code level, analysts consider each line of code and each variable within that line of code. In FMEA at the method or class level, analysts consider transactions among software elements in terms of variables exchanged among them. The task is to determine system consequences when each variable has an unexpected value – regardless of system states. Very robust code will catch many errors caused by unexpected values and work around them, or at least ignore them. On the other hand, loss of some data due to such failures may also have serious consequences. Software FMEA identifies these failures in terms of variables and the expected consequences.

Since software FMEA is concerned with system consequences without regard to cause, it takes the review process a step beyond traditional code reviews because code reviews are based on the seemingly reasonable assumption that “software doesn’t fail.” Values in registers and values of pointers can change unexpectedly. If a single incidence of an unexpected value can cause a catastrophic failure, then a FMEA can identify where

code should be made more robust to detect and gracefully handle this kind of failure.

FMEA will also identify code weaknesses that may cause less severe consequences.



Chapter 5

Integrative Method of FTA and Software FMEA for Security Analysis

5.1 Integrating forward and backward analysis

Regarding to comprehension above, the strength of forward analysis (identifying previously unknown failure modes) and the strength of backward analysis (identifying combinations of events and circumstances that could cause the hypothesized fault to occur) are complementary [\[46-48\]](#).

Thus, some current views regarding the limited effectiveness of forward analysis were not supported by the results of integrated forward and backward analysis approach.

This thesis describes our use of a forward search method, Software FMEA, followed by a backward search method somewhat similar to FTA, to assist in analyzing the software requirements for critical portions of the spacecraft software [\[49\]](#).

5.2 Forward Integrated Security Analysis of SFMEA and FTA

The strength of forward analysis (identifying previously unknown failure modes) and the strength of backward analysis (identifying combinations of events and circumstance that could cause the hypothesized fault to occur) are complementary. Thus, some current views regarding the limited effectiveness of forward analysis were not supported by the results of our integrated forward and backward analysis approach [\[52\]](#). Accordingly, security analysis which involves SFMEA was taken as the main concepts

for the forward integrated for this security analysis of smartphone, decompose by FTA as supplementation. The mechanism of forward integrated software security analysis is depicted in the **Figure 5-1**. Then, we will classify the differences both of forward and backward analysis in the areas of Smartphone availabilities. Forward analysis principle implies SFTA can be performed in sequence according to the severity degree of failure effects from the results of SFMEA. Also, the weakness of this method is the failure modes can be used as intermediate events of SFTA to identify the causes of the failure modes, which might be difficult to identify or express with SFMEA. Security analysis can be showed more comprehensively through forward integrated analysis. The merits within this method, improvement actions might be suggested [12][31].

Software FTA is able to evaluate in many sequence related to the severity degree of the failure effects on smartphone from the results of software FMEA generator. The cause and effect with higher severity ought to be implemented as top events of software FTA to classify the causes of these effect details [50-51]. Also, the failure mode can use as intermediate events if software FTA to identify the causes of the failure modes, which should be hard to identify or explain with Software FMEA, security analysis can be showed more integrative through forward integrated analysis as shown below.

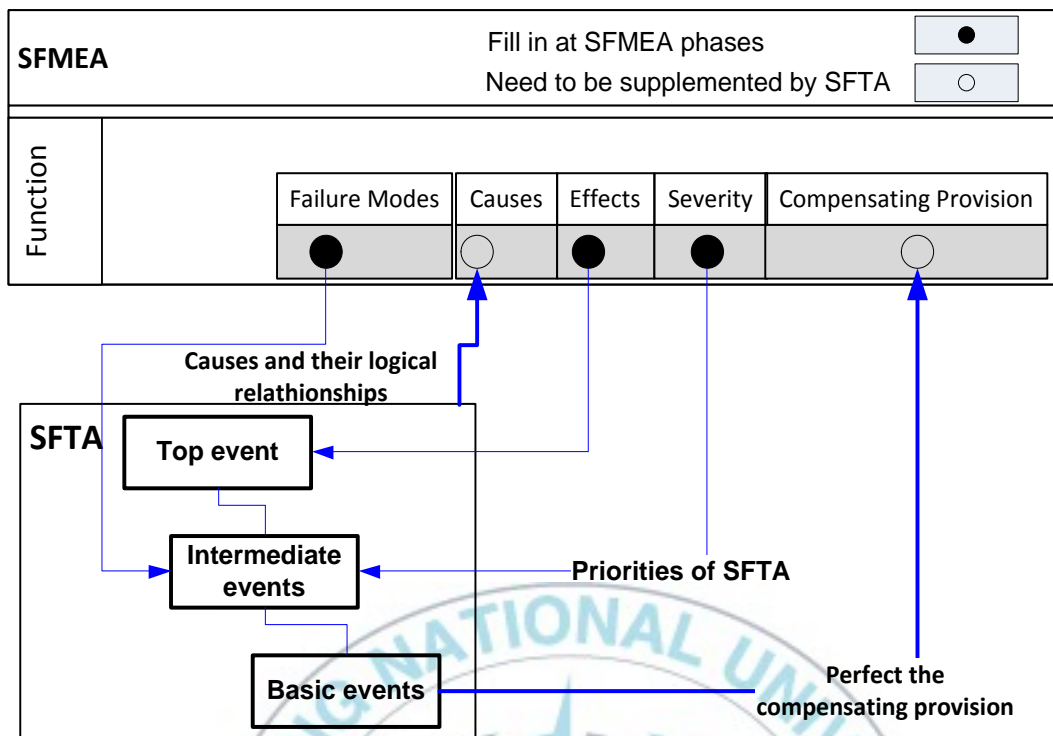


Figure 5-1: Forward Integration Security Analysis Technique of Smartphone Failure for SFMEA

Here the step of the forward integrated analysis methods.

- 1) First step: Web can choose analysis level for security software for smartphone, both of functional neither structural level within the device.
- 2) Second step: Evaluation of this failure will be performed to the matrix of each failure effects. If condition permitted, the critical analysis may be performed by multiplying the value of severity, occurrence and detection, a risk priority number can be defined. Severity is the FMEA which show the seriousness of the effect of the failure. Occurrence is the FMEA which show the frequency of the failure. And detection is the probability of the failure being detected of the impact effect.
- 3) Step third: the higher severity degree as top event is function of security analysis SFTA for smartphone.

- 4) Step fourth: Intermediate event for software security analysis on the smartphone is like erratic, freeze failure.
- 5) Step Fifth: The failure mode for this device such as alive event, low battery event, reboot and manual off fault, which will figured on the basic FTA, See on the **Figure 5-1.**

Forward search has capabilities to identify unexpected data or behavior that can cause the failure modes.

5.3 Backward Integrated Security Analysis of SFTA and SFMEA

The backward is similar to a FTA, except that the root node (the cause) is not necessarily of a fault or even an event. A FTA, on the other hand, takes a known fault or hazard as its root and works backward to determine the possible causes. Sometime backward analysis is applied to code, whereas the backward analysis here is applied to software requirements.

However, FTA has been recommended for use during requirements analysis to check security constraints. Since FTA has been extensively described elsewhere, no further description is provided in this thesis. SFTA is taken as the main concept for the backward integrated security analysis, followed by SFMEA as supplementation, the principle of backward integrated analysis is simulated in **Figure 5-2.**

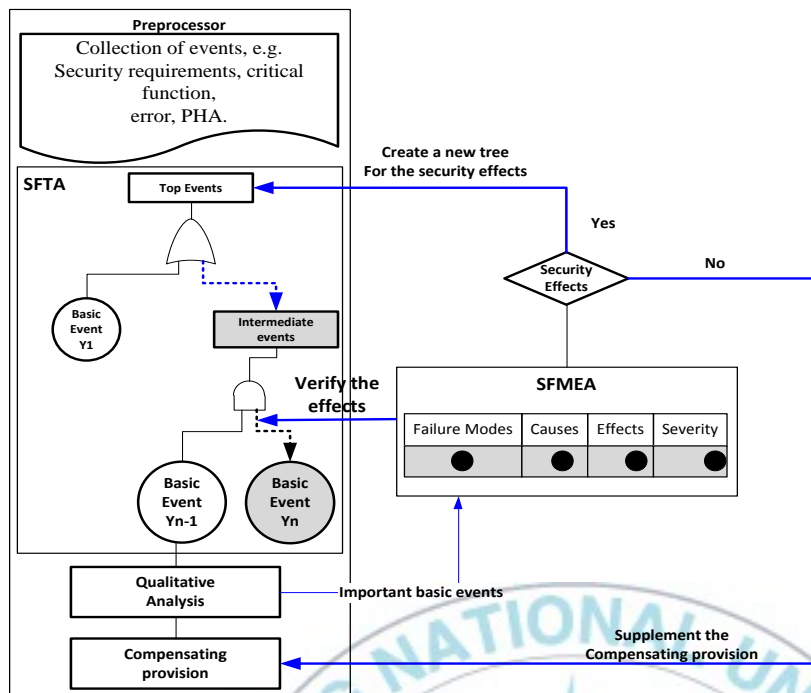


Figure 5-2: Backward integrated security analysis technique of Smartphone failure for SFMEA

Select the undesired events as top events to construct software FMEA security analysis fault trees. The selection of top events might be security requirements from development documentation, the most significant bottom events identified by qualitative analysis of Software FTA, is able to use as failure modes to perform Software security FMEA to verify the failure clue, if incorrectness in fault tree is identified, amendment might be taken to update the fault tree. In the other hand, new top events should emerge according to the severity degree of failure effects, because one failure mode might lead several consequences on the software system [12][34].

If criticality analysis has been performed in the process of software security analysis FMEA, the occurrence probability of the top event can be calculated with the assistant of report. The backward integrated security analyses on the smartphone fault technique are performed.

- First step: Collecting the top events base on software security requirement, and create the software fault trees.
- Second step: Classifying the minimal cut-sets and necessary level of bottom events are, the lower the order of minimal cut-sets. More necessary this minimal cut-sets, the bottom events in minimal cut-set with lower order is more important than that with higher order, the most often a bottom event appears in different minimum cut-set, the most important that bottom event, if the orders of the minimal cut-set is the same.
- Third step: Performing software security analysis of SFMEA with the more important bottom events taken as failure modes.
- Fourth step: Redesigning the software security analysis of the fault trees and development actions.
- Fifth step: Classifying new failure effects and causes as top events to build new fault trees and deploy security analysis furthers.
- Sixth step: Create the failure of effect, cause and severity to create new symbols of failure mode.

Backward search has common capabilities to analyze enabling circumstances contributing to possibility of unexpected data or behavior.

5.4 Event Name Populations

Based on the bussiness process management which we describe through analysis system above. However, the populations data related to the fault, failure and error inside smartphone failure analysis, we can consider main point related to this research :

- Smartphone failure of the security analysis

This failure event is the main point within this analysis, first event or TOP event in this failure is to discovery failurity within smartphone, thus, the TOP level on the building of the fault tree analysis with the subject is smartphone failurity as depicted on the Figure 5-3 below. Smartphone is a smart device with numerous application inside the system of it [41], on the other hand, with the collecting data analysis which already classified through **Table 5.1**, we can cosider to built a faul tree analysis system based on this calculation process, however, by generating data value within this failure which have been transformed by the bussiness process management as depicted on the **Figure 1-1**. We can classify the sub main failure as is below :

- Hardware, software and database (DB) Failure
- Logger architecture failure
- Hardware and software in severity

Most of the these failure activities have been supported by classifying object failure entities in the minimal cut sets which we populated in to the **Table 5.1**.

Table 5.1 : Event Name Failure for Smartphone Security Analysis

| No. | Event Name | Q mean | Failures | FVImp | RDF | RIF | FC | Sensitivity Value |
|------|-----------------------------|----------|----------|----------|----------|---------|-----------|-------------------|
| 0.0 | Smartphone failure | 0.003150 | | | | | | |
| 1.0 | Hardware failure | 0.000040 | | | | | | |
| 2.0 | Software failure | 0.000123 | | | | | | |
| 3.0 | Database failure | 0.000240 | | | | | | |
| 4.0 | Logger architecture failure | 0.000350 | | | | | | |
| 5.0 | Hardware and software in | 0.001350 | | | | | | |
| | System engineering | | | | | | | |
| 1.1 | fault inside HW | 0.004560 | | | | | | |
| 1.2 | Architecture fault inside | 0.003420 | | 0.000423 | 1.084584 | 458.764 | 0.000541 | 9.87880 |
| 1.3 | Architecture bugs | 0.000340 | | 0.000535 | 1.083434 | 458.764 | 0.000562 | 8.64321 |
| 1.4 | Delay/not real time | 0.000450 | | 0.000434 | 1.000012 | 458.764 | 0.000566 | 7.03414 |
| 1.5 | Assembly HW Fault | 0.000234 | | 0.000342 | 1.000432 | 458.764 | 0.000566 | 1.63657 |
| 1.6 | Architecture unreliable | 0.000115 | | 0.000324 | 1.000342 | 458.764 | 0.000566 | 2.56565 |
| 1.7 | Multimedia failure | 0.000121 | | 0.000432 | 1.003445 | 458.764 | 0.000565 | 2.45464 |
| 2.1 | Freezing and halting | 0.000124 | | | | | | |
| 2.2 | Self-shutdown and silent | 0.000125 | | 0.000231 | 1.003446 | 451.153 | 0.000565 | 1.56229 |
| 2.3 | Erratic failure(unstable | 0.000128 | | 0.000221 | 1.003476 | 25.527 | 0.000536 | 1.30253 |
| 2.4 | Output value failure | 0.000128 | | 0.000232 | 1.003432 | 262.266 | 0.000257 | 1.03233 |
| 2.5 | Input output omission fault | 0.000129 | | 0.000256 | 1.003478 | 456.247 | 0.000527 | 1.00000 |
| | Ring or music volume | | | | | | | |
| 2.6 | fault(misconfigurations) | 0.000127 | | 0.000267 | 1.003421 | 456.324 | 0.000457 | 1.32636 |
| 2.7 | Charge indicator | 0.000125 | | 0.000287 | 1.003432 | 659.217 | 0.000557 | 1.56562 |
| 2.8 | Software registry records | 0.000128 | | 0.000289 | 1.003434 | 568.000 | 0.000460 | 1.00000 |
| 2.9 | Kernel and OS service | 0.000132 | | 0.000290 | 1.003421 | 251.000 | 0.000458 | 1.56560 |
| 2.10 | Software security is | 0.000143 | | 0.000234 | 1.003478 | 695.326 | 0.000853 | 1.03112 |
| 2.11 | Building package fault | 0.000112 | | 0.000214 | 1.003487 | 548.232 | 0.000857 | 1.03266 |
| 2.12 | Home screen fault | 0.000121 | | 0.000254 | 1.003432 | 954.566 | 0.000589 | 1.23656 |
| 2.13 | Web system error | 0.000111 | | 0.000267 | 1.003443 | 26.363 | 0.000589 | 1.20000 |
| 2.14 | Kernel system fault | 0.003420 | | 0.004896 | 1.003432 | 65.626 | 0.000875 | 1.00000 |
| 3.1 | DB engine file | 0.003123 | | | | | | |
| 3.2 | DB engine registry is | 0.003423 | | 0.004843 | 1.003657 | 485.327 | 0.000588 | 1.22356 |
| 4.1 | Heartbeat technique | 0.002345 | | | | | | |
| 4.2 | freeze and self-shutdown | 0.002453 | | 0.004687 | 1.003979 | 451.253 | 0.000875 | 1.00000 |
| 4.3 | Bluetooth is | 0.001234 | | 0.004892 | 1.003343 | 546.213 | 0.000875 | 1.23235 |
| | Unreliable, neglectable, | | | | | | | |
| 4.4 | forget required | 0.003241 | | 0.004890 | 1.003426 | 654.627 | 0.0008756 | 1.23203 |
| 4.5 | Bias in result | 0.002345 | | 0.004892 | 1.003855 | 235.000 | 0.0008756 | 1.56562 |
| | Crashing detection (| | | | | | | |
| 4.6 | heartbeat AO event fault) | 0.001234 | | 0.004894 | 1.003647 | 455.124 | 0.000875 | 1.20003 |
| | Low battery indicator | | | | | | | |
| 4.7 | (battere status failure) | 0.001123 | | 0.004897 | 1.003736 | 654.656 | 0.000876 | 1.03062 |
| 4.8 | fault event) | 0.001123 | | 0.004876 | 1.006563 | 658.254 | 0.000875 | 1.00256 |
| | Reboot incident(shutdown, | | | | | | | |
| 4.9 | halting event) | 0.001212 | | 0.004842 | 1.007237 | 654.899 | 0.000876 | 1.02150 |

| | | | | | | | |
|--|----------|--|----------|----------|---------|----------|---------|
| MAOF incident(user deliberated 4.10 turn off fault) | 0.001214 | | 0.004898 | 1.007427 | 954.569 | 0.000876 | 1.03603 |
| Panic detector failure (application 4.11 and collecting data fault) | 0.001235 | | 0.004878 | 1.004326 | 657.231 | 0.000786 | 1.02542 |
| 4.12 Data logging engine fault | 0.005434 | | 0.004856 | 1.007273 | 234.563 | 0.000786 | 1.03265 |
| 4.13 Power management fault | 0.006757 | | 0.004845 | 1.002728 | 123.214 | 0.000786 | 1.06357 |
| 4.14 Shutdown fault | 0.007869 | | 0.004835 | 1.006727 | 342.232 | 0.000898 | 1.65650 |
| 5.1 Severity in high detections | 0.008997 | | | | | | |
| 5.2 Severity in medium detections | 0.007878 | | 0.004868 | 1.007628 | 326.232 | 0.000876 | 1.05402 |
| 5.3 Severity in low detections | 0.005236 | | 0.004896 | 1.006273 | 654.237 | 0.000786 | 1.65656 |

However, to calculate sensitivity value, we try to calculate the limitation number recorded by Table 5.1 above on the section of sensitivity value. On the other hand, sensitivity value is sensitivity analysis allowing the automatic variation of event failure and repair data between specified limits [23][42-43]. On this study, we try to simulate limitation number such as are, if limits number “1.5” the icon set will be green, if then limits number between “1.5” and “8” the icon set will be brown, and if final limits number is greater than “8” the icon set will be red, which means that the number almost close to the “10” (sensitivity factor) [44]. Further regarding to the explanation it is depicted by **Figure 5-3** below.



Figure 5-3: Legend of sensitivity value

5.5 Integrative Methods of FTA and SFMEA for Smartphone Security Analysis

In this section, we will have some guidelines in the selection of integrated analysis techniques. First is forward integrative security analysis, an analysis which works in this phase should be comprehensive and meticulous enough to discover software defects within smartphone as completely as possible at early stage of software development. Sometimes this method might be a better choice to avoid omission due to human factors, error correcting code, and so on. Second is backward integrated security analysis. Commonly backward integrated analysis is a technique which advocates efficiency. It might be feasible to select the undesired events with higher severity degree or of greater concern to carry out analysis in the design phase within smartphone fault and failure. Furthermore, we create cut sets by approaching FTA method for system failure within smartphone structured; who is the higher level severity is fault in the system software. In the **Figure 5-4** the process to minimize the high level failure will be performed.

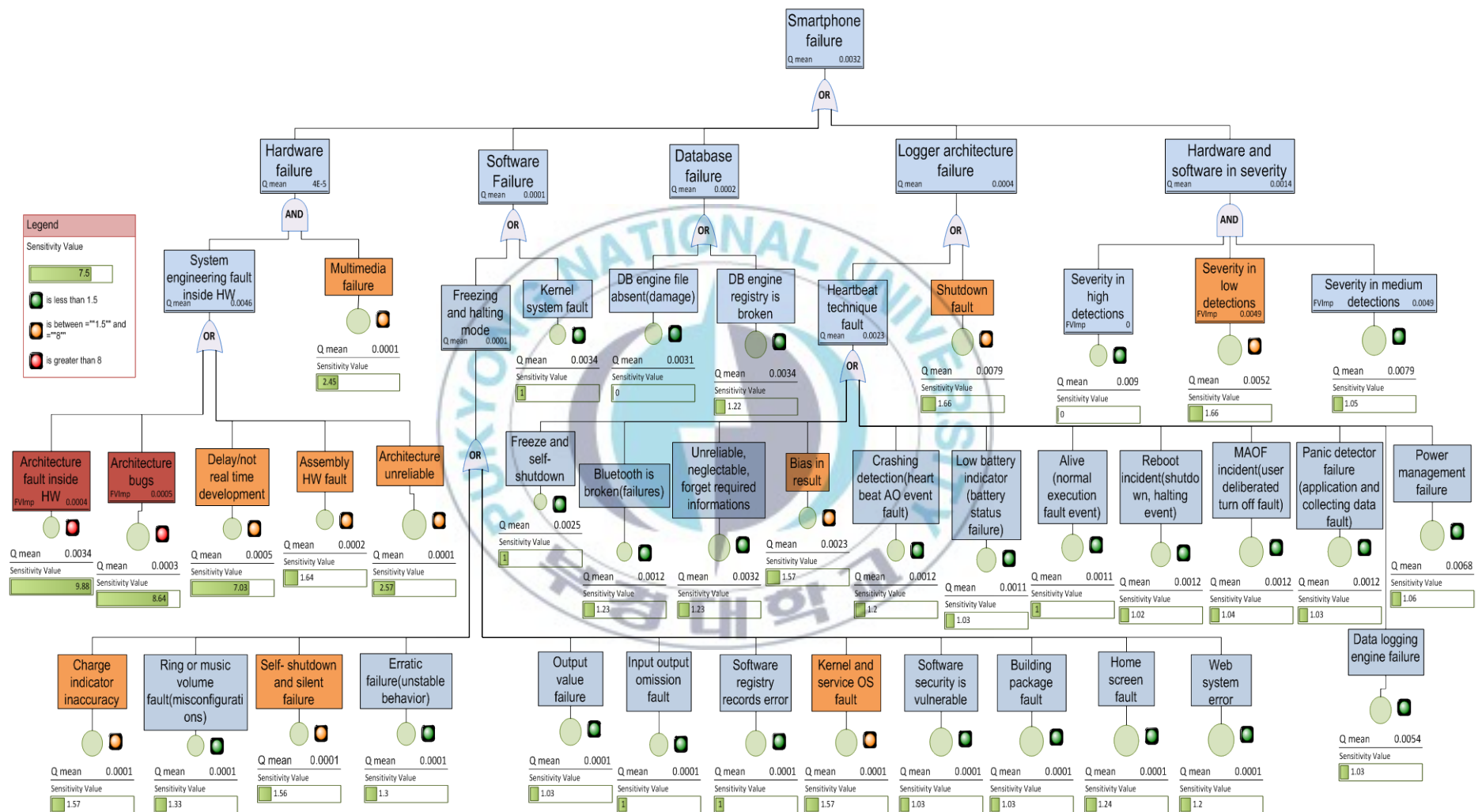


Figure 5-4: Fault Tree Analysis (FTA) for Smartphone Failure

Based on the **Figure 5-4** above, we can mention both of the security analysis within Smartphone, integrated method analysis in requirement and integrated method analysis in framework. So, each of the 45 functional modules is analyzed thoroughly and 35 failures mode are identified. Failure modes with higher severity degree of failure effects are also identified, e.g. crash detection, shutdown failure, ring and music volume misconfiguration, inaccuracy charge indicator fault, failure on behavior, Bluetooth failure, power management failure, freeze, and so on. We can use these failures effects as top events to perform FTA. Thus, integrated method for security analysis within this framework, use this failure effect as a top event to build a new fault tree for further analysis, automatically a new improvement action, adding a new software watchdog module, is recommended to activate software reset function when hardware reset signal is shielded, after framework is modifying to the improvement tools, software FTA continues based on supplementation of new deployment information. A new bottom event, namely, watchdog failure is detected, which is then used as a failure mode to perform software FMEA.

On the other hand, we have already extracted all the fault tree system in to the fault tree consideration which is depicted in to the **Figure 5-4**: fault trees of the Smartphone on the software failure, within this analysis we have found 3 (three) major(s) of the critical system; they are self-shutdown and silent failure, kernel and service OS fault, and charge indicator inaccuracy. Most of the three categories is major capacity by using value is between =”1.5” and =”8” within sensitivity value as depicted by using **Figure 5-5** below. During the analysis, rest of the three majors who we found ahead, it is concluded of the minor’s categories; with the value number is less than 1.5.

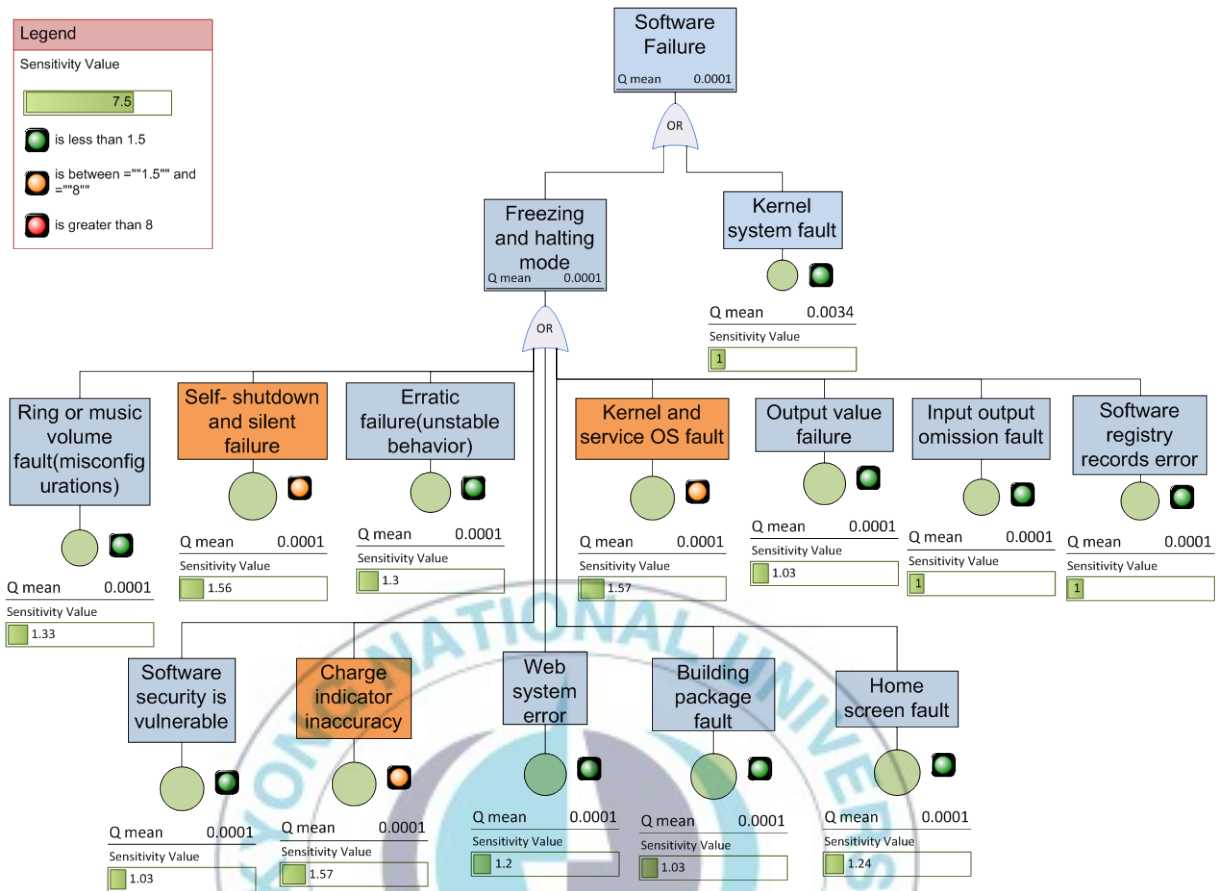


Figure 5-5: Fault Trees of the Smartphone on the Software Failure

Based on this FTA system on the software failure categories, we have already populated numerous of the entities failure, by using the inductive TOP event failure in the software failure within Smartphone. However, we can understand the process steps taken related to the Smartphone with the cases of the software failure by performing the **Figure 5-5** above.

Furthermore, based on the **Figure 5-6**, which is described the fault trees for the smartphone on the logger architecture failure; easily we can perform with existing two major(s) categories we found, they are shutdown fault with Q mean value are 0.0079 and sensitivity value is 1.66, then, a couple of bias in result with Q mean value are

0.0023 and sensitivity value is 1.57. For detail fault tree, it is depicted on the **Figure 5-6** below.

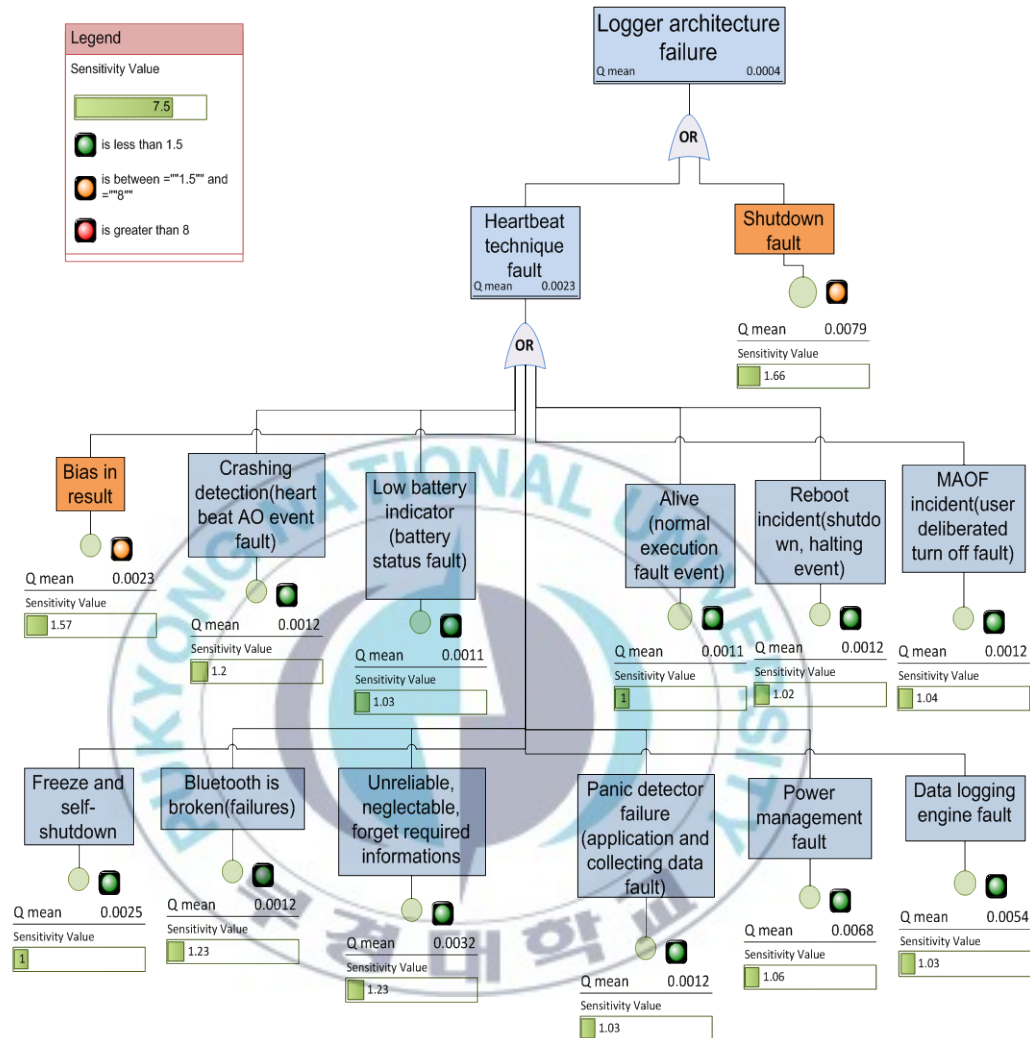


Figure 5-6: Fault Trees of the Smartphone on the Logger Architecture Failure

By analyzing this fault tree system, most of the inductive TOP event of the logger architecture failure is minor condition, well-known minority of the failure within this event, so, we can say this device event is norm conditionally.

Next, on the inductive TOP event within hardware failure as shown in the **Figure 5-7** below, most of the event in “critical” conditions, we already understand between major capacities which is illustrated by brown events and critical capacities which is illustrated

by red events, so, we can conclude that there is two categories within the value number on the critical analysis, they are architecture fault inside hardware with the Q mean value number is 0.0034, failure value impact is 0.0004 and the sensitivity value is 9.88 or “critical” condition, then, on the event of architecture bugs with the failure value impact is 0.0005, while Q mean value is 0.0003, then sensitivity value is 8.64 or “critical” condition indeed. Thus, for detail analysis, it is already depicted by **Figure 5-7** below.

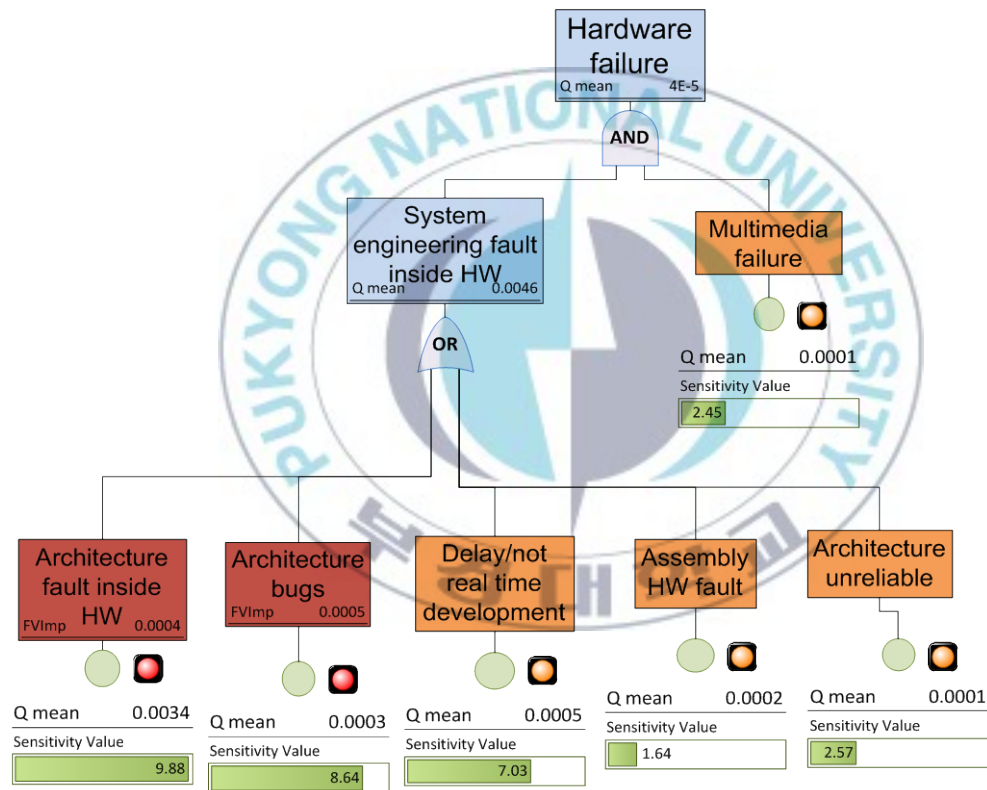


Figure 5-7: Fault Trees of the Smartphone on the Hardware Failure

However, within smartphone failure analysis, we also created for inductive TOP event in the database failure. Part of this event is very important to detect a faults, error and misconfiguration function in the smartphone application system. As depicted in the **Figure 5-8** fault trees of the smartphone on the database failure, we can conclude that

this event device is “normal” conditions as shown in the **Figure 5-8**, this analysis value is using the earlier edition of the failure within smartphone report value [8].

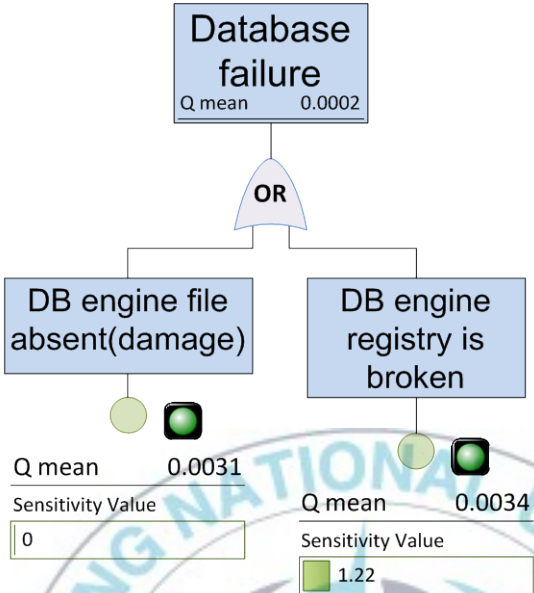


Figure 5-8: Fault Trees of the Smartphone on the Database Failure

Finally, we have already generated the business process related to the smartphone failure events, thus, we can calculate the all event in to consideration, by using this fault tree system, the rest of the analysis failure out the top event as depicted by **Figure 5-9** below.

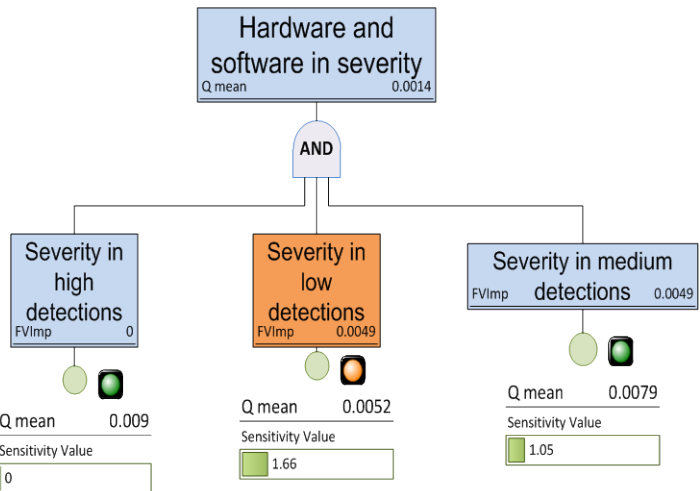


Figure 5-9: Fault Trees of the Smartphone within Hardware and Software in Severity

Accordingly, fault trees of the smartphone inside hardware and software in severity is part of the security system within smartphone application, both of the hardware and software is very important thing to support this application running. Hardware system related to the upgrading of the memory, ROM, RAM, integrated circuit, network facilities, and so on. This device is influenced by real time application upgrading conduct. Otherwise, software system is followed by the hardware requirement, if hardware is upgrading then software will be upgrading too. By using this fault tree system, actually we can show based on the **Figure 5-9** that the major report is found in severity in low detection, it means that the tools to detect is unreliable to defrag, on the other hand, we already find the critical within the tools with numerous value such as Q mean is 0.0052, failure value impact is 0.0049 and sensitivity value is 1.66 that means major conditions.

However to describe step taken activities regard to figure above, we will analyze activity steps into consideration procedures of the combining template based on FTA, such as;

- (1) Step first, Identification of hazard and related failure: First, the undesired event of the system is determined and the corresponding output in the failure is identified. Next, all networks that can contribute to this output are identified.
- (2) Step second, Fault tree generation, the top event template is put at the top of the fault tree with the undesired event as the top event, and the templates of the blocks directly connected to the output block are attached. Each branch is expanded until there are no dependent failure routines or function blocks left. When all templates are attached, terminal node templates are added to the

remaining cause nodes that are leaf nodes of the generated tree. ‘variable/value’ templates are attached to the value or variable cause node whereas the comparator/operator’ templates are attached to the operator or comparator nodes. When expanding fault trees, analyst may choose to simplify the fault tree by eliminating irrelevant branches. For example, if an unwanted event occurred at the output of an AND block by outputting an incorrect value 1, only the right most sub tree.

- (3) Step third, Cut-set analysis, the last step is to generate the minimal cut sets, as typically performed on security analysis, so that analysts may obtain additional insights as to how logical design errors found through fault tree analysis can be best corrected.

Thus, it needs to cross checking of FTA and FMEA into consideration. With this analysis, we give a procedure for the combined template based on fault tree analysis with a case study applied on a smartphone failure design as summarized in analyze activity steps above, and we draw the fault tree diagram as depicted in the **Figure 5-4** to **Figure 5-9**. FMEA method is an analysis which emphasizes successful functions rather than potential failures and dangers. It allows for controlling a product to operate without failure within a limited time, or to operate a product failure free for a set period of time in between errors [45]. FMEA is effective for adding an additional failure free event by analyzing a single unit or single failure. The comprehensive analytic logical approach for security analysis based on FMEA is depicted by the **Figure 5-10**. The SFMEA process was found to be successful in identifying some ambiguous,

inconsistent, and missing requirements. More importantly, the SFMEA process, followed by a backward analysis somewhat similar to FTA, identified five significant and unresolved requirements issues, they are hardware failure, software failure, database failure, logger architecture failure and HW and SW in severity.

The potential failure effect is that software abort abnormally event out of control. Thus, an improvement action which is performed through **Table 5.2** will be classified within failure mode effect analysis framework. Furthermore, we identify some items for this failure effect both of software application and hardware system. See detail the table below for more explanations.

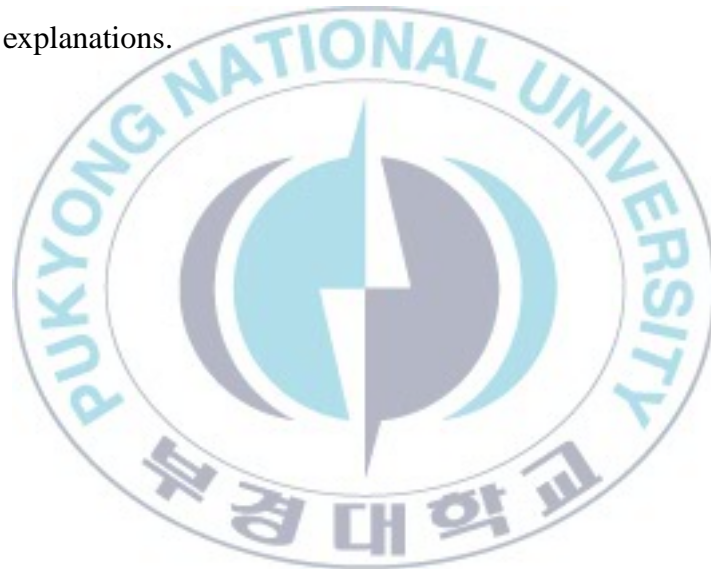


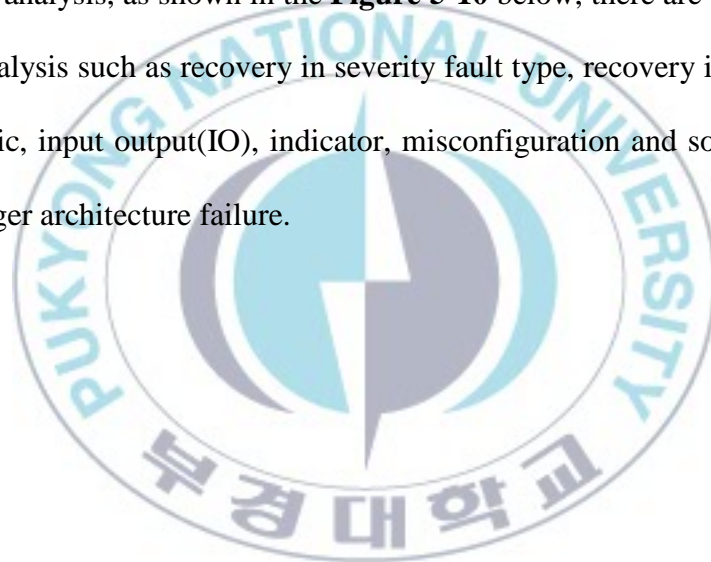
Table 5.2: Worksheet of the SFMEA Module for Security Analysis on the Smartphone

| ID | Item | Failure Mode | Causes | Effects | Severity | Compensating provision |
|----|------|--|-------------------|--------------------------|-----------|---|
| 18 | SW | Device output constant, not respond to user's input | Freeze | Lock-up, halting failure | High | Repeat the action; Mean Time Between Freeze(MTBFr, heartbeat) |
| | | Device self-shutdown, no service delivered to user interface | Self-halt | Silent Failure | Minor | Wait an amount of time; Mean Time Between Self Shutdown (MTBFr-Shutdown) |
| | | Device exhibits erratic without any input inserted by the user, e.g. backlight flashing, self-activation. | Unstable behavior | Erratic Failure | Moderate | Reboot(Power cycle or reset); Running Application Detector; |
| | | Error monitoring and failure data analysis, fault injection and design methodology | Weak security | Failure data analysis | Moderate | Shifting error sources, explosive complexity, and global volume inaccurate. |
| | | Failure data for data logger applications | Software Failure | Out of date | High | Software Event Failure detector; Rtimer Updater |
| 19 | HW | The device, in response to an input sequence, delivers an output sequence that deviates from expected one. E.g. inaccuracy in charge indicator, ring or music volume different from configuration. | Output Failure | Value Failure | Very High | Remove Battery; Power Manager action |
| | | User inputs have no effect on device behavior,e.g. soft keys do not work | Input Failure | Omission failure | High | Service the phone both of reset software and repair the hardware |
| | | Failure data for Bluetooth distributed system | Hardware failure | Out of date | Hazardous | Panic event failure; Panic detector action |

Based on the references of the **Table 5.2** above, that is a kind of worksheet of the SFMEA module for security analysis on the Smartphone, we can construct the backward integrated of SFMEA in to the recovery system within smartphone. Hence, numerous of the cause and effect of the smartphone can classify with severity groups

such as freeze and lock-up or halting failure (high), self-halt and silent failure (minor), unstable behavior and erratic failure (moderate), weak security and failure data analysis (moderate), software failure and out of date (high), output failure and value failure (very high), input failure and omission failure (high), and hardware failure and out of date (hazardous) [26].

However, to inducing the integrated recovery smartphone fault by using backward integrated of SFMEA security analysis. We use 3 (three) AND-gate and 1 (one) OR-gate within this analysis, as shown in the **Figure 5-10** below, there are 4 (four) events in the recovery analysis such as recovery in severity fault type, recovery in hardware fault, recovery in logic, input output(IO), indicator, misconfiguration and software fault, and recovery in logger architecture failure.



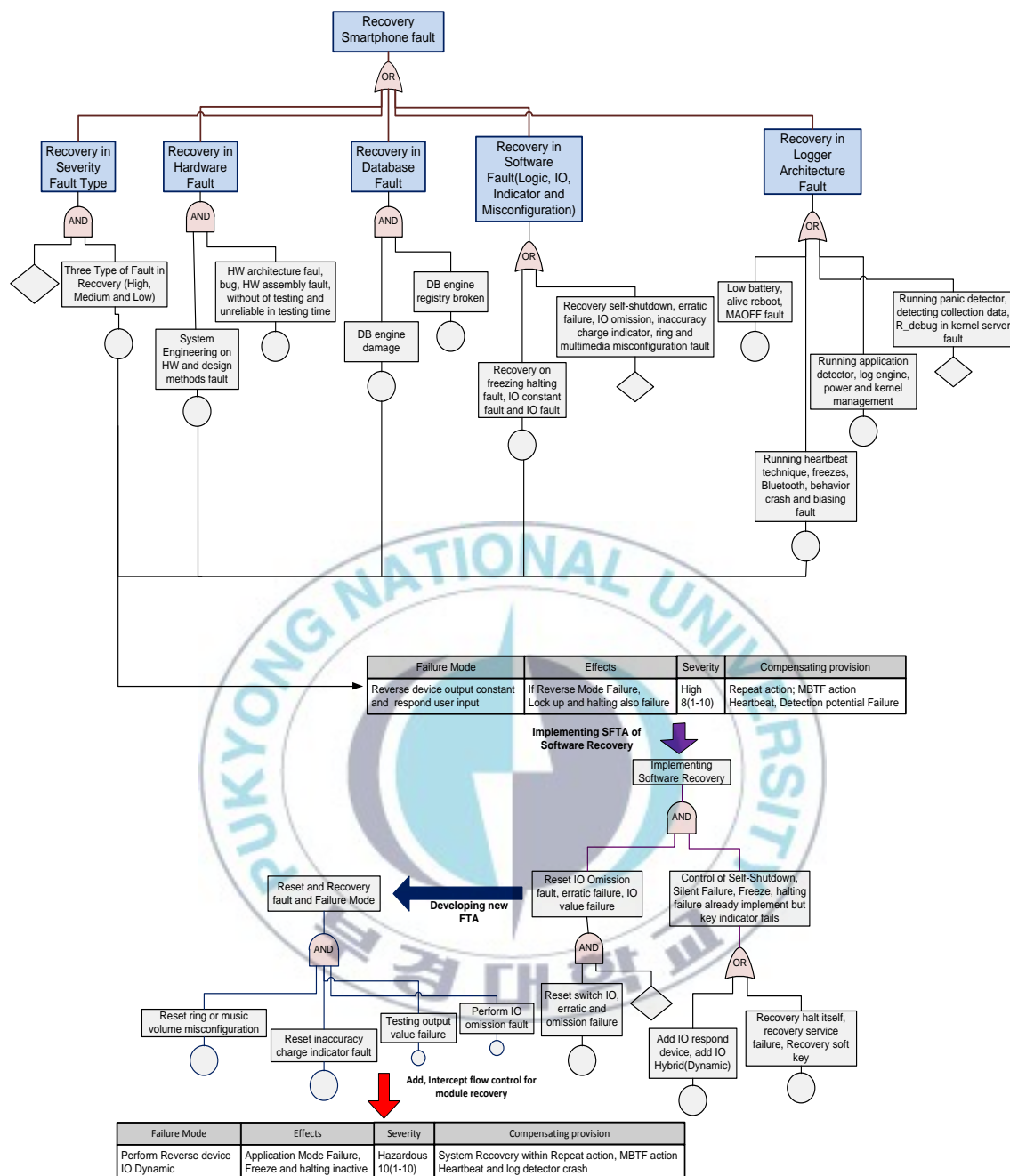


Figure 5-10: Backward Integration of the Smartphone Security Analysis

SFTA and SFMEA can be applied as supplementary techniques for each other. More comprehensive and effective analysis result can be obtained by integrating both of this method. On the other hand, these two analysis can be integrated become single view.

Meanwhile, characteristic integration performed different directions. It looks to be efficient and more reliable for backward analysis. While, forward method for analyzing human error and detect the software both of hardware completely at framework task. Thus, to perform data percentage from RPN, the RPN is equal multiplication from severity, occurrence and detection. Then, it will be figured on the **Table 5.3** below.

Table 5.3: Data Population for Cut-sets of Risk Priority Number (RPN) within Smartphone Security Analysis

| ID | Item | Failure Modes | Severity (S) | Occurrence (O) | Detection (D) | RPN (SOD) |
|----|------|--|--------------|----------------|---------------|-----------|
| 18 | SW | Device output constant, not respond to user's input | 8 | 7 | 6 | 336 |
| | | Device self-shutdown, no service delivered to user interface | 9 | 6 | 8 | 432 |
| | | Device exhibits erratic without any input inserted by the user, e.g. backlight flashing, self-activation. | 6 | 7 | 5 | 210 |
| | | Error monitoring and failure data analysis, fault injection and design methodology | 6 | 5 | 8 | 240 |
| | | Failure data for data logger applications | 9 | 8 | 6 | 432 |
| | | Σ Software Fault | 38 | 33 | 33 | 1650 |
| 29 | HW | The device, in response to an input sequence, delivers an output sequence that deviates from expected one. E.g. inaccuracy in charge indicator, ring or music volume different from configuration. | 10 | 9 | 7 | 630 |
| | | User inputs have no effect on device behavior, e.g. soft keys do not work | 8 | 7 | 9 | 504 |
| | | Failure data for Bluetooth distributed system | 10 | 6 | 8 | 480 |
| | | Σ Hardware Fault | 28 | 22 | 24 | 1614 |

For further recovery, we conduct to take action within software security variable as shown by using **Table 5.4** below.

Table 5.4: Software Security Variable on Recovery Action

| Entity | Failure Type | Recovery/Security Action | | | | | |
|----------|------------------------------|--------------------------|-------------------------------|-----------------|------------------------|--------|------------|
| | | Service Phone | Reboot(Power cycle or reset) | Battery Removal | Wait an amount of time | Repeat | Unrepeated |
| Hardware | Freeze | | √ | √ | | | |
| | Input Failure | √ | | | | | √ |
| | Output Failure | | √ | | | | |
| | Self-Shutdown | | √ | √ | | | |
| | Unstable Shutdown | | | √ | √ | | |
| Software | Failure Severity | | √ | | | | |
| | Heartbeat | | √ | | | √ | |
| | Running Application Detector | √ | | | | | √ |
| | Log Engine | | √ | | √ | √ | |
| | Power Manager | √ | | | | | |
| | Panic Detector | √ | | | √ | √ | √ |

By collecting data from recovery Smartphone failure analysis table, thus, we can consider to build the matrix table in order performance of the recovery failure analysis is able to represent on the real matrix analysis table, so the **Table 5.5** is the answer for resume the failure analysis in to data real time.

Table 5.5: Risk Matrix Analysis for Smartphone's Security Analysis

| High Level (HL) Event | Panic Category | Applications | | | | | | | | | | | | | | | | | |
|-----------------------|----------------|--------------|---------|----------|-------------|----------------------|-------|-----------|-----|--------------|------------------------|----------|--------------------|---------|----------|-----------|----------|-----------|--------|
| | | Log Browser | Browser | Messages | Message Log | Camera Log Telephone | Clock | Clock Log | Log | Log Contacts | BT_Browser Log Teleph. | Contacts | Telephone Contacts | Battery | Messages | Telephone | Explorer | Clock Log | TomTom |
| Freeze | KERN - EXEC | 1 | | | 0 | | 1 | 1 | 1 | 1 | 1 | | | | | 0 | 1 | 1 | |
| Self-Shutdown | KERN - EXEC | | 1 | | | 1 | | | 0 | | | | 0 | 0 | | | 1 | | 0 |
| | MSGSG Client | 0 | | 1 | | 0 | | | 1 | | | 0 | | | | | 0 | | |
| No HL event | E32USER-Cbase | | | | | | 0 | | | 1 | | | | 1 | | | | 0 | |
| | EIKC OCTL | | 1 | | | 1 | | | | 0 | | | | | | | 1 | | |
| | EIKON-LISTBOX | | | 0 | | | | 1 | | | | | 1 | 1 | 0 | 1 | | | 0 |
| | KERN - EXEC | | 1 | | 1 | | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | USER | 1 | | 1 | | | 1 | | | | | 1 | | 0 | | 1 | 0 | | 1 |
| | Viewsrv | 1 | | | | 1 | 1 | | 1 | 1 | 1 | | | | | | | 1 | F |

Legend:

T = True [Logic number 1]

F = False [Logic number 0]

5.6 FTA, SFMEA and FMECA as Security System

As a security system, either of SFTA, SFMEA and SFMECA. In particularly, most of them are created to be reliable system within security system on the software development process, hence, below we classify this approach to be **Table 5.6** shown.

Table 5.6: FTA versus FMECA Selection Criteria

| Selection Characteristics | Preferred | |
|---|------------------|--------------|
| | FMEA | FMECA |
| Security of public/operating/maintenance personnel | ✓ | |
| Small number/clearly defined TOP events | ✓ | |
| Indistinctly defined TOP events | | ✓ |
| Full-Mission completion critically important | ✓ | |
| Many, potentially successful missions possible | | ✓ |
| All possible failure modes are of concern | | ✓ |
| High potential for human error contributions | ✓ | |
| High potential for software error contributions | ✓ | |
| Numerical risk evaluation needed | ✓ | |
| Very complex system architecture/many functional paths | ✓ | |
| Linear system architecture with little human/software influence | | ✓ |
| System irreparable after mission starts | ✓ | |



Chapter 6

Conclusions and Future Work

In this thesis, we propose integrative methods of SFTA and SFMEA for security analysis of a Smartphone, on this integrative approach both of software FTA and FMEA for security analysis; the tool will generate a secure simple approaching algorithm to keep a reliable for software safety and security within smartphone as our basic reason of the research. The analysis reporting is performed through a global sales figures and market share with 3rd quarter, software security issue incorporated through operating system, functional flow diagram for software security requirement analysis, on the other hand, the variable of smartphone fault related to software security system, threat attack model especially for mobile device security, a logic functional block diagram with three aspects diagram functional likewise, software, hardware and data logger which is supported by five key aspect of security, threat model of security, three cause of target security, forward and backward integrated security analysis technique inside of software SFTA and SFMEA, business process related to security mode, the main basic algorithm of SFTA as secure analysis approaching, worksheet of the SFMEA module of security analysis, a more comprehensive and effective analysis through backward integrated analysis algorithm within SFTA and SFMEA, a matrix approaching formula, the software security variable recovery for failure, a logical board such as integrity security mechanism both of the tools, and finally the matrix analysis performance evaluation tables. We believe this concept is a reliable integrative method in software security analysis through integration approaching within SFTA and SFMEA. Thus, this

paradigm is a high quality for software safety and security analysis with smartphone failure object as the research and development areas of study.

In these methods and technologies, SFTA, which generates the use cases by the minimal cut-sets of fault trees, can't determine the priorities of all the use cases and can't utilize the finished smartphone analysis security test result. In order to solve these problems, a security analysis approach with SFMEA which are transferred from fault trees is described above through matrix production. FTA is an important verification methodology for security analysis. As a top down technique, FTA can be used to analyze the origin of the failure, determine the security requirements, detect the logic errors, identify the multiple failure sequences involving different parts of the system (such as hardware, human, and software), and guide the security test. In the software security testing, FTA can be used to determine appropriate input data for testing, and detail the test case definition for the sets of validation test cases to be executed.

On the future work, we can mention that failure on the smartphone is very complex application, and this opportunity is good passion for the scientist and researcher to maintain within research and development on the smartphone, either on the software, hardware and network features.

However, SFTA and SFMEA is method and technologies, that use a cut-set analysis value, this tool is very simple and elegant, that way, many scientist and researcher is conducting to develop this system in numerous applications nowadays. Within this thesis, we create a pseudo code to build the FTA system both of symbol neither of connector line. In this source code we use Visual Basic programming language which integrated to the Microsoft Visio for designing the tools. Inside of the source code we deploy business process requirement for construct the tool is to the real connection, in

particular in this source code, we use MVC (model view controller) for dynamic connection and programming flow as shown by using **Table 6.1** below.

Table 6.1: Pseudo Code to develop FTA Cut-Sets by Using Visual Basic Programming Language

```

Public Sub EnumerateRules()
'declare the object oriented on the first preparation process
Dim doc As Visio.Document
Dim ruleSet As Visio.ValidationRuleSet
Dim rule As Visio.ValidationRule
Dim datObj As DataObject
Dim txt As String
Set doc = Visio.ActiveDocument
txt = "EnumerateRulesSets Count = " & doc.Validation.RuleSets.Count

For Each ruleSet In doc.Validation.RuleSets
If ruleSet.Enabled Then

txt = txt & vbCrLf & "EnumerateRules for RuleSet : " & _
ruleSet.nameu & " : Count = " & ruleSet.Rules.Count
txt = txt & vbCrLf & "ID" & vbTab & "Category" & vbTab & "NameU" & vbTab & _
"Description" & vbTab & "TargetType" & vbTab & _
"FilterExpression" & vbTab & "TestExpression"

For Each rule In ruleSet.Rules
With rule

txt = txt & vbCrLf & .ID & vbTab & .category & vbTab & .nameu & vbTab & _
.description & vbTab & .targettype & vbTab & _
.filterexpression & vbTab & .testexpression
End With
Next
End If
Next
Set datObj = New DataObject
datObj.SetText txt
datObj.PutInClipboard
End Sub

```

```

'Adding a Rule Set
Public Sub AddRuleSet()
' Add a validation rule set to the document.
' Edit the nameU to suit.
Dim doc As Visio.Document
Dim ruleSet As Visio.ValidationRuleSet
Dim nameu As String
nameu = "Fault Tree Analysis"
Set doc = Visio.ActiveDocument
' Check whether the rule set already exists.
Set ruleSet = getRuleSet(doc, nameu)
If ruleSet Is Nothing Then
' Create the new rule set.
Set ruleSet = doc.Validation.RuleSets.Add(nameu)
End If
ruleSet.description = "Fault Tree Analysis rule set."
ruleSet.Enabled = True
ruleSet.RuleSetFlags = Visio.VisRuleSetFlags.visRuleSetDefault
End Sub

```

```

Private Function getRuleSet(ByVal doc As Visio.Document, _

```

```

    ByVal nameu As String) As Visio.ValidationRuleSet
' Return a named rule set or nothing.
Dim retVal As Visio.ValidationRuleSet
Dim ruleSet As Visio.ValidationRuleSet
    Set retVal = Nothing
    For Each ruleSet In doc.Validation.RuleSets
        If UCase(ruleSet.nameu) = UCase(nameu) Then
            Set retVal = ruleSet
        Exit For
    End If
Next
    Set getRuleSet = retVal
End Function

```

```

Public Sub DeleteRuleSet()
' Delete a rule set from the active document.
' Edit the ruleSetNameU value to suit.
'Deleting a Rule Set
'We can use the following DeleteRuleSet() code procedure to delete an unwanted rule in a rule set.
Dim doc As Visio.Document
Dim nameu As String
    nameu = "Fault Tree Analysis"
    Set doc = Visio.ActiveDocument
' Check whether the rule set already exists.
    If Not getRuleSet(doc, nameu) Is Nothing Then
' Delete the rule set.
        doc.Validation.RuleSets.Item(nameu).Delete
    End If
End Sub

```

```

Public Sub AddRule()
' Add a rule to named rule set.
' Edit nameU and the addARule arguments to suit.
Dim doc As Visio.Document
Dim ruleSet As Visio.ValidationRuleSet
Dim nameu As String
    nameu = "Fault Tree Analysis"
    Set doc = Visio.ActiveDocument
' Check whether the rule set already exists.
    Set ruleSet = getRuleSet(doc, nameu)
    If ruleSet Is Nothing Then
        Exit Sub
    End If
' Add the rule.
    addARule ruleSet, "Connectivity", "UngluedConnector", _
        "Connector is not glued at both ends.", 0, _
        "ROLE()=1", _
        "AND(AGGCOUNT(GLUEDSHAPES(4)) = 1, AGGCOUNT(GLUEDSHAPES(5)) = 1)"
End Sub

```

```

Private Function getRule(ByVal ruleSet As Visio.ValidationRuleSet, _
    ByVal nameu As String) As Visio.ValidationRule
' Return a named rule or nothing.
Dim retVal As Visio.ValidationRule
Dim rule As Visio.ValidationRule
    Set retVal = Nothing
    For Each rule In ruleSet.Rules
        If UCase(rule.nameu) = UCase(nameu) Then
            Set retVal = rule
        Exit For
    End If
Next
    Set getRule = retVal
End Function

```

```

Private Sub addARule(ByVal ruleSet As Visio.ValidationRuleSet, _
    ByVal category As String, ByVal nameU As String, _
    ByVal description As String, ByVal targettype As Integer, _
    ByVal filterexpression As String, ByVal testexpression As String)
' Add or update a validation rule in the document.
Dim rule As Visio.ValidationRule
Set rule = getRule(ruleSet, nameU)
If rule Is Nothing Then
    Set rule = ruleSet.Rules.Add(nameU)
End If
rule.category = category
rule.description = description
rule.Ignored = False
rule.targettype = targettype
rule.filterexpression = filterexpression
rule.testexpression = testexpression
' Flush existing issues to ensure re-validation.
ruleSet.Document.Validation.Issues.Clear
End Sub

```

```

Public Sub DeleteRule()
' Delete a rule from a rule set.
' Edit ruleSetNameU and ruleNameU to suit.
Dim ruleSetNameU As String
Dim ruleSet As Visio.ValidationRuleSet
ruleSetNameU = "Fault Tree Analysis"
Set ruleSet = getRuleSet(Visio.ActiveDocument, ruleSetNameU)
If ruleSet Is Nothing Then
    Exit Sub
End If

```

```

Dim rulenameU As String
rulenameU = "UngluedConnector"
' Check whether the rule already exists.
If Not getRule(ruleSet, rulenameU) Is Nothing Then
' Delete the rule.
ruleSet.Rules.Item(rulenameU).Delete
End If
End Sub

```

```

' Adds rules to named rule set from Excel.
' To use this in a VBA project, add a reference to the "Microsoft Excel 14.0 Object Library".
Public Sub AddRulesFromExcel()

    Dim xlWorkbook As Excel.Workbook
    Dim doc As Visio.Document
    Dim ruleSet As Visio.ValidationRuleSet
    Dim nameU As String
    nameU = "Fault Tree Analysis"

    Dim category As String
    Dim rulenameU As String
    Dim description As String
    Dim targettype As String
    Dim filterexpression As String
    Dim testexpression As String

    Const categoryCol As Integer = 2
    Const rulenameCol As Integer = 3
    Const descriptionCol As Integer = 4
    Const targettypeCol As Integer = 5
    Const filterexpressionCol As Integer = 6
    Const testexpressionCol As Integer = 7

    Set doc = Visio.ActiveDocument

```

```

' Check whether the rule set exists already.
Set ruleSet = getRuleSet(doc, nameU)
If ruleSet Is Nothing Then
    Exit Sub
End If

' Get data from Excel.
' Assumes you have created an Excel spreadsheet at the path shown, that contains the rules you want to add,
' and with columns that correspond to the constants declared in this subroutine.
Dim xlApp As New Excel.Application

Set xlWorkbook =
xlApp.Workbooks.Open("C:\Users\semilab\Documents\FTASmartphone\FTASmartphone.xlsx ")

On Error GoTo AddRulesFromExcel_Err

Dim xlWorkSheet As Excel.Worksheet
Set xlWorkSheet = xlWorkbook.Worksheets(1)

Dim numRows As Integer
numRows = xlWorkSheet.UsedRange.Rows.Count

' Assumes that a header row exists and skips over it.
For xlRow = 2 To numRows

    category = xlWorkSheet.Cells(xlRow, categoryCol)
    rulenameU = xlWorkSheet.Cells(xlRow, rulenameCol)
    description = xlWorkSheet.Cells(xlRow, descriptionCol)
    targetType = xlWorkSheet.Cells(xlRow, targetTypeCol)
    filterexpression = xlWorkSheet.Cells(xlRow, filterexpressionCol)
    testexpression = xlWorkSheet.Cells(xlRow, testexpressionCol)

    addARule ruleSet, category, rulenameU, description, targetType, filterexpression, testexpression

Next xlRow

AddRulesFromExcel_Err:
If (Err.Number) Then
    Debug.Print Err.description
End If

xlWorkbook.Close
xlApp.Quit

End Sub

```

The *FilterExpression* value checks the layer assignment and verifies that a transfer symbol shape is to be tested. The *TestExpression* value checks that there is only one glued connector.

| | |
|-------------------------|---|
| <i>Category</i> | Connectivity |
| <i>NameU</i> | Transfer |
| <i>Description</i> | Transfer symbol shape should have one connector. |
| <i>TargetType</i> | 0 |
| <i>FilterExpression</i> | AND(ONLAYER("Flowchart"),STRSAME(LEFT(MASTERNAME(750),15),"Transfer symbol")) |
| <i>TestExpression</i> | AGGCOUNT(GLUEDSHAPES(0))= 1 |

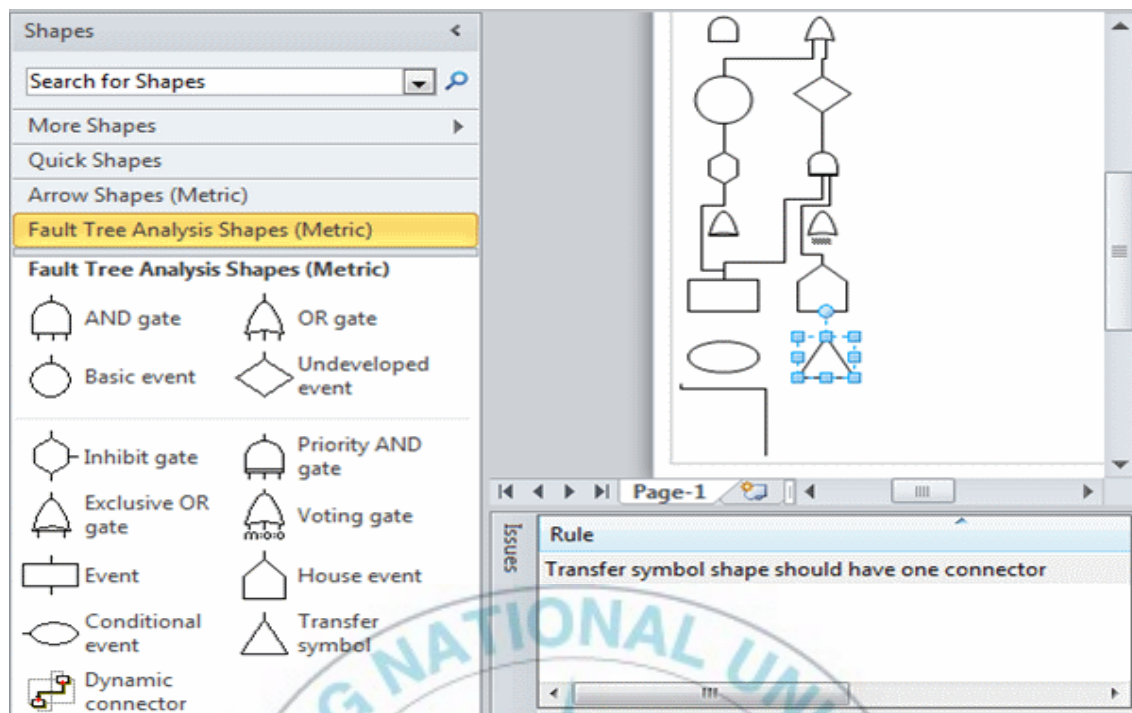


Figure 6-1: Developing the transfer symbol shape should have one connector

References

- [1] Marcello Cinque, Domenico Cotroneo, Zbigniew Kalbarczyk, Ravishankar K. Iyer, "How Do Mobile Phones Fail? A Failure Data Analysis of Symbian OS Smartphones," in Proceedings of 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '07), pp.585-594, 2007.
- [2] A. Shabtai, Y. Fledel, Elovici, "Automated Static Code Analysis for Classifying Android Applications Using Machine Learning," in Proceedings of 2010 International Conference on Computational Intelligence and Security (CIS), pp.329-333, 2010.
- [3] H. Reza, S. Buettner, V. Krishna, "A Method to Test Component Off-the-Shelf (COTS) Used in Safety Critical Systems," in Proceedings of 5th International Conference on Information Technology: New Generations 2008(ITNG08), pp.189-194, 2008.
- [4] Jules White, Chris Thompson, Hamilton Turner, Brian Dougherty, Douglas C. Schmidt, "Primary Title: Wreck Watch: Automatic Traffic Accident Detection and Notification with Smartphones," in Proceedings of Mobile Networks and Applications, Springer Netherlands, vol.16, pp.285-303, 2011.
- [5] Hsiu-Sen Chiang, Woei-Jiunn Tsaur, "Identifying Smartphone Malware Using Data Mining Technology," in Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN2011), pp.1-6, 2011.
- [6] M. Sullivan, R. Chillarege, "Software Defects and Their Impact on System Availability - A Study of Field Failures in Operating Systems," in Proceedings of 21st International Symposium on Fault-Tolerant Computing (FTCS-21), pp.2-9, 1991.
- [7] Yuan Wei, Jin Qin, "Safety-Driven Software Reliability Allocation in Medical Device Application," in Proceedings of 2011 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), pp.105-109, 2011.
- [8] Kumar Maji, A. Kangli Hao, S. Sultana, S. Bagchi, "Characterizing Failures in Mobile OSes: A Case Study with Android and Symbian," in Proceedings of

- IEEE 21st International Symposium on Software Reliability Engineering (ISSRE), pp.249-258, 2010.
- [9] Retrieved on 10th may 2012 from <http://www.gartner.com/it/page.jsp?id=1848514>.
- [10] A. Gopalan, S. Banerjee, A.K. Das, S. Shakkottai, "Random Mobility and the Spread of Infection," In Proceedings of IEEE of INFOCOM 2011, pp.999-1007, 2011.
- [11] B. S. Medikonda, P. S. Ramaiah, "Integrated Safety Analysis of Software-Controlled Critical Systems," ACM Transaction on Software Engineering, IEEE Computer Society, vol.35, No.1, Nov. 2010.
- [12] T. Maier, "FMEA and FTA to Support Safe Design of Embedded Software in Safety-Critical Systems," In Proceedings of CSR 12th Annual Workshop on Safety and Reliability of Software Based Systems, Bruges, Belgium, 1995.
- [13] M. Becher, F.C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, C. Wolf, "Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices," in Proceedings of IEEE Symposium on Security and Privacy (SP), pp.96-111, 2011.
- [14] Ashwin Chaugule, Zhi Xu, Sencun Zhu, "A Specification Based Intrusion Detection Framework for Mobile Phones", Lecture Notes in Computer Science, Applied Cryptography and Network Security, vol. 6715, pp.19-37, 2011.
- [15] R. Mojdehrahsh, Wei-Tek Tsai, S. Kirani, L. Elliott, "Retrofitting Software Safety in an Implantable Medical Device," Journal of IEEE Software, vol.11, no.1, pp.41-50, Jan. 1994.
- [16] Zhang Hong, Liu Binbin, "Integrated Analysis of Software FMEA and FTA," In Proceedings of International Conference on Information Technology and Computer Science (ITCS09), vol.2, pp.184-187, 2009.
- [17] M. Becher, "Security of Smartphones at the dawn of their ubiquitousness," Ph.D. dissertation, University of Mannheim, Oct. 2009.
- [18] Shuaifu Dai, Yaxin Liu, Tielei Wang, Tao Wei, Wei Zou, "Behavior-Based Malware Detection on Mobile Phone," In Proceedings of 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM10), pp.1-4, 2010.

- [19] N.G. Leveson, P.R. Harvey, "Analyzing Software Safety," IEEE Transactions on Software Engineering, vol.SE-9, no.5, pp. 569- 579, September 1983.
- [20] C. J. Price, "Function Directed Electrical Design Analysis", Artificial Intelligence in Engineering, vol. 12(4), pp. 445–456, 1998.
- [21] Wildan Toyib, Man-Gon Park, "Process Analysis of Digital Right Management for Web-Based Multicast Content," Journal of Korean Multimedia Society, vol. 14, no.12, pp.1601-1612, Dec. 2011.
- [22] Retrieved on 05th May 2012 from <http://www.albservice.com/en/reliability-products/fta.html>
- [23] Retrieved on 08th May 2012 from http://www.isograph-software.com/_techspecs/psa32techspec.pdf
- [24] F. Balbastro, "Fault Tolerant Systems for Mobile Devices," Laboratory for Advanced Software System, Dec. 2006.
- [25] Myong-Hee Kim, Man-Gon Park, "A Study on the Software Fault Modes and Effect Analysis for Software Safety Evaluation," Journal of Korean Multimedia Society, vol. 15, no.1, pp.115-130, Jan. 2012.
- [26] C.F. Kemerer, B.S. Porter, "Improving the Reliability of Function Point Measurement: An Empirical Study," IEEE Transactions on Software Engineering, vol.18, no.11, pp.1011-1024, 1992.
- [27] M. Zitser, R.P. Lippmann, T. Leek, "Testing Static Analysis Tools Using Exploitable Buffer Overflows From Open Source Code," In Proceedings of ACM Sigsoft 2004/FSE Foundations of Software Engineering Conference, 2004.
- [28] M. A. Zhivich, T. Leek, R. P. Lippmann, "Dynamic Buffer Overflow Detection Tools," In Proceedings of Defining the State of the Art in Software Security Tools Workshop, NIST Special Publication 500-264, Eds. P. Black and E. Fong, National Institute of Standards and Technology, pp. 95–101, August 10, 2005.
- [29] K. Kratkiewicz, "Evaluating Static Analysis Tools for Detecting Buffer Overflows in C Code," Master's Thesis, Harvard University, Cambridge, Massachusetts, March 2005.
- [30] Wayne Jansen, Karen Scarfone, "Guidelines on Cell Phone and PDA Security," In Proceedings of National Institute of Standard and Technology, US Department of Commerce, pp.800-124, 2008.

- [31] P. Zheng, L.M. Ni, "Spotlight: The Rise of the Smartphone," In Proceedings of IEEE Distributed Systems Online, vol.7, no.3, 2006.
- [32] W. W. Streilein, K. Kratkiewicz, K. Piwowarski, S. Webster, "PANEMOTO: Network Visualization of Security Situational Awareness through Passive Analysis," In Proceedings of 8th Annual IEEE SMC Information Assurance Workshop, West Point, New York, June 20-22, 2007.
- [33] Tao Jianfeng, Wang Shaoping, Yao Yiping, "Hybrid Method of Computer Aided FMECA and FTA," Journal of Beijing University of Aeronautics and Astronautics, Beijing, China, vol.26, no.6, pp. 663-665. 2000.
- [34] R. K. Cunningham, M. Zhivich, "Securing Process Control Systems of Today and Tomorrow," In Proceedings of Critical Infrastructure Protection Conference, Hanover, NH. Mar. 19-21, 2007.
- [35] K. Kratkiewicz, R. Lippmann, "A Taxonomy of Buffer Overflows for Evaluating Static and Dynamic Software Testing Tools," In Proceeding of Workshop on Software Security Assurance Tools, Techniques, and Metrics, NIST Special Publication 500-265, Eds. P. E. Black, M. Kass and E. Fong, National Institute of Standards and Technology, pp. 44-51, 2005.
- [36] K. Kratkiewicz, R. Lippmann, "Using a Diagnostic Corpus of C Programs to Evaluate Buffer Overflow Detection by Static Analysis Tools," In Proceedings of Defining the State of the Art in Software Security Tools Workshop, NIST Special Publication 500-264, Eds. P. E. Black and E. Fong, National Institute of Standards and Technology, pp. 102-111, 2005.
- [37] T. Leek, R. Lippmann, M. Zitser, "Testing Static Analysis Tools Using Exploitable Buffer Overflows From Open-Source Code," Foundations of Software Engineering, Newport Beach, California, 31 October – 5 November 2004.
- [38] Young Mo Kang, Chanwoo Cho, Sungjoo Lee, "Analysis of Factors Affecting the Adoption of Smartphones," In Proceedings of 2011 IEEE International Conference on Technology Management (ITMC), pp.919-925, 2011.
- [39] S. Schechter, A. Ozment, "Milk or Wine: Does Software Security Improve with Age?" In Proceeding of USENIX Security 2006, Vancouver, British Columbia, 31 July 2006.

- [40] S. Schechter, A. Ozment, "The Security of Open BSD: Milk or Wine? Login" In Proceedings of Advanced Computing Systems, Berkeley, California, Dec, 23th 2006.
- [41] Y. Huang, C. Kintala, N. Kolettis, N.D. Fulton, "Software Rejuvenation: Analysis, Module and Applications," In Proceedings of the 23rd International Symposium on Fault-Tolerant Computing FTCS-23, pp.381-390, 1995.
- [42] T.R. Leek, G.Z. Baker, R.E. Brown, M.A. Zhivich, R.P. Lippman, "Coverage Maximization using Dynamic Taint Tracing," MIT Lincoln Laboratory TR-1112, March 2007.
- [43] R.E. Brown, R.I. Khazan, M.A. Zhivich, "AWE: Improving Software Analysis through Modular Integration of Static and Dynamic Analyses," Program Analysis for Software Tools and Engineering, San Diego, CA, June 13-14, 2007.
- [44] Alan C. Tribble, Steven P. Miller, "Software Safety Analysis of a Flight Management System Vertical Navigation Function - A Status Report," In Proceedings of IEEE the 22th Digital Avionics Systems Conference, Indianapolis, CA, pp.12-16, 2003.
- [45] Y. Huang, C. Kintala, "Software Implemented Fault Tolerance: Technologies and Experience," In Proceedings of the 1993 International Symposium on Fault-Tolerant Computing, pp. 2-9, 1993.
- [46] D. Dagon, T. Martin, T. Starner, "Mobile Phones as Computing Devices: The Viruses are coming!" In Proceedings of IEEE on Pervasive Computing, vol.3, no.4, pp. 11- 15, 2004.
- [47] Tansu Alpcan, Christian Bauckhage, Aubrey-Derrick Schmidt, "A Probabilistic Diffusion Scheme for Anomaly Detection on Smartphones," Lecture Notes on Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices, pp.31-46, 2010.
- [48] B. Scorgie, P. Veeraraghavan, S. Ghosh, "Early virus detection for Windows Mobile," In Proceedings of 9th Malaysia International Conference on Communications (MICC2009), pp.295-300, 2009.
- [49] Robyn R. Lutz, Robert M. Woodhouse, "Requirements analysis using forward and backward search," Journal of Annals of Software Engineering, vol.3, no. 1, pp. 459-475, 1997.

- [50] A. Shabtai, Y. Fledel, Y. Elovici, "Securing Android-Powered Mobile Devices Using SELinux," In Proceedings of IEEE Security & Privacy, vol.8, no.3, pp.36-44, 2010.
- [51] S. Chandra, P.M. Chen, "Whither Generic Recovery from Application Faults? Fault Study Using Open-Source Software," In Proceedings of International Conference on Dependable Systems and Networks (DSN 2000), pp.97-106, 2000.
- [52] Lutz, Robyn and Woodhouse, Robert, "Requirements analysis using forward and backward search," Journal of Annals of Software Engineering, Springer Netherlands, vol.3, pp.459-475, 1997.



Acknowledgement

(감사의 말씀)

It has been indeed a unique experience of my life to study for Master degree at Pukyong National University, Busan, and Republic of Korea. Today when I am completing my thesis and happy of my achievements, I feel as it was a dream to be here with so many good memories which I would treasure in rest of my life. I enjoyed every moment of my study in PKNU for several reasons. The PKNU environments, the available facilities and above all, the faculty, staff and student all are at their best and future base oriented.

I owe the credit of my study and achievements to my advisor Prof. Dr. Man-Gon Park, who played a pivotal role in persuading me to enter to his department to research in the field of Software Engineering and Multimedia Information Processing Laboratory. He facilitated me in numerous ways throughout my stay at Busan. I admire his knowledge and wisdom. His brief and to the point guidance has been a strong force for me to push my work to the best standards. Prof. Dr. Man-Gon Park is one of the exceptional and extraordinary persons to study with. From the very deep of my heart, I am thankful to Prof. Man-Gon Park for everything, he did for me during my study in PKNU. Then special thank is achieved to Mr. Ari Satria, MA., Commercial of attaché for Republic of Korea, Indonesia embassy at Seoul.

I have been fortunate as a foreign student getting special attention from the professors in the department. Thus, how can I forget the special attention and care extended to me by Prof. Dr. Kyung-Hyune Rhee and Prof. Dr. Chang-Soo Kim. They have been very kind to me to give me special coaching time during the formal defense. I learnt a lot from them. The special dinners extended to me by Prof. Dr. Kyung-Hyune Rhee and Prof. Dr. Man-Gon Park will remain a source of happiness for me as long as trip in Jogjakarta [Malioboro Street] and Borobudur Indonesia. They treated me like their colleague which made me a distinctive and privileged person in the department. It would be unfair if I do not acknowledge the support and prayers, I got from my family.

My parents have been a source of my strengths as their non-stop prayers made me more

powerful and successful all the time. I would like to especially acknowledge my wife for taking care of and other family affairs during my long absence from home. Without her support, it would not have been possible for me to complete my Master research which required my full attention with peace of mind. Her continuously prayed for my success and health and kept reporting me of their successes in their studies and other matters which made me happy and satisfied all the time. My sisters, elders and little brothers have been a source of my spiritual strength as them kept me in these prayers all the time and looked after my family during my absence. I wish them healthy and prosperous life. Thank you to all my Indonesian friends either worker immigrant and students, This thesis would not have been possible without the support from A/Prof. Dr. Myong Hee Kim, all the Semi-Lab members such as Mrs. Kim Gyu Ah, Ms. Jin Eun-ji, Mr. Kim Sung-cul, Mr. Shin Hyun-jin, Mr. Kim Dong-gyun, Ms. Yun Seo-yun, Ms. Kim So-young, Ms. Sul Min-bi, Mr. Sang Twang, and Mr. Irwin Chea. I enjoyed all weathers of Busan including hot summer, winter, autumn and flowery spring. I will miss Korean food with which I am so used to now. I would definitely be happy and look forward in the future, for the opportunities to come to Korea and visit PKNU and meet my Korean Professors and friends.

During the last days of thesis writing, I got very sad news of the demise of my father. He was a very pious and prayerful person who always kept me in his prayers to complete this study. Unfortunately I could not pass this information of completion of my master thesis to him in his life. I dedicate my master thesis to my little angel Ms. Azqanaira Rizqi Wilani Anindya [Alwa], who was born during the thesis writing.