

The copyright © of this thesis belongs to its rightful author and/or other copyright owner. Copies can be accessed and downloaded for non-commercial or learning purposes without any charge and permission. The thesis cannot be reproduced or quoted as a whole without the permission from its rightful owner. No alteration or changes in format is allowed without permission from its rightful owner.



**AN INVESTIGATION OF REQUIREMENTS TRACEABILITY  
PRACTICES IN SOFTWARE COMPANIES IN MALAYSIA**

**JASIM MOHAMMED DAHR**

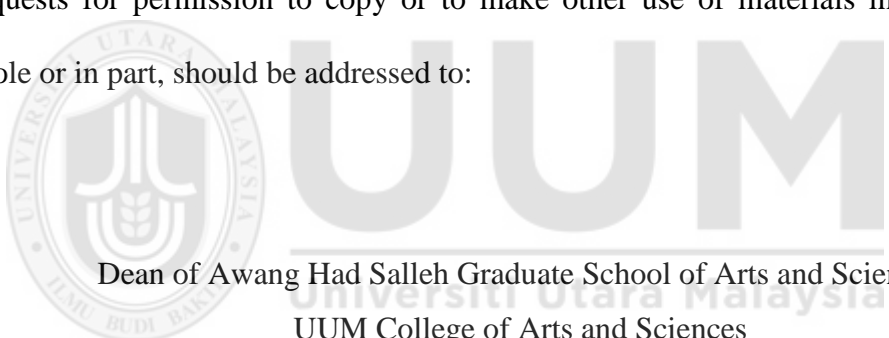


**MASTER OF SCIENCE (INFORMATION TECHNOLOGY)  
SCHOOL OF COMPUTING  
UUM COLLEGE OF ARTS AND SCIENCES  
UNIVERSITI UTARA MALAYSIA  
2016**

## **Permission to Use**

In presenting this dissertation in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:



Dean of Awang Had Salleh Graduate School of Arts and Sciences  
UUM College of Arts and Sciences

Universiti Utara Malaysia

06010 UUM Sintok

## **Abstrak**

Keupayaan mengesan keperluan (RT) merupakan salah satu aktiviti yang kritikal dalam menguruskan keperluan dengan baik dan bahagian yang penting dalam pembangunan projek. Pada masa yang sama, RT dapat meningkatkan kualiti produk perisian. Walau bagaimanapun, pengamal industri sukar melaksanakannya memandangkan kekurangan garis panduan atau hasil yang dapat memberi panduan kepada mereka dalam melaksanakan RT dalam projek mereka secara berkesan. Ini menunjukkan pengamal industri kekurangan maklumat tentang cara yang terbaik atau paling berkesan dalam melaksanakan tugas mereka, seperti di syarikat perisian. Walaupun begitu, terdapat beberapa amalan umum yang diterima pakai yang dapat memberi panduan kepada pengamal industri dalam menjejaki keperluan projek mereka. Kajian ini bertujuan untuk menentukan amalan RT melalui sorotan kajian sistematik. Selain itu, kajian ini telah menjalankan soal selidik untuk mengkaji penggunaan amalan RT di syarikat-syarikat perisian di kawasan utara Malaysia. Akhir sekali, satu siri temu bual dengan pengamal industri telah dijalankan untuk mengetahui sebab-sebab yang mempengaruhi kepada penggunaan amalan ini dalam pembangunan perisian. Dapatan kajian menunjukkan bahawa majoriti syarikat perisian tidak menggunakan amalan RT untuk mengesan keperluan kerana isu kewangan dan kekurangan pengetahuan tentang amalan RT ini. Kajian ini membentangkan bukti empirikal berkaitan penggunaan amalan RT antara syarikat perisian. Di samping itu, kajian ini membolehkan pengamal industri untuk mencari kaedah yang sesuai untuk mengesan keperluan, dan juga membolehkan penyelidik untuk mencari jurang dan petunjuk untuk kajian masa hadapan dalam kajian domain ini.

Kata kunci: Amalan Keupayaan Mengesan Keperluan, Teknik Keupayaan Mengesan, Alatan Keupayaan Mengesan, Sorotan Kajian Sistematik

## **Abstract**

Requirement traceability (RT) is one of the critical activity of good requirements management and an important part of development projects. At the same time, it improves the quality of software products. Nevertheless, industrial practitioners are challenged by this lack of guidance or results which serve as a rule or guide in establishing effective traceability in their projects. The outcome of this is that practitioners are ill-informed as to the best or most efficient means of accomplishing their tasks, such as found in software companies. Notwithstanding the lack of guidance, there are a number of commonly accepted practices which can guide industrial practitioners with respect to trace the requirements in their projects. This study aims to determine the practices of RT through conducting a systematic literature review. Also, this study conducted a survey for investigating the use of RT practices in the software companies at northern region of Malaysia. Finally, a series of interviews with practitioners were carried out to know the reasons that influence on the use of these practices in software development. The findings showed that majority software companies do not use traceability practices for tracing requirements due to financial issues and the lack of knowledge of these practices. This study presented empirical evidence about the use of RT practices among software companies. Thus, the findings of this study can assist practitioners to select RT practices, and also enables researchers to find gaps and pointers for future study in this study domain.

**Keywords:** Requirements Traceability Practices, Traceability Techniques, Traceability tools, Systematic Literature Review

## **Acknowledgement**

In the Name of Allah, the Most Gracious and Most Merciful. First of all, I would like to thank Allah (SWT), for having made everything possible by giving me strength, confidence, and courage to accomplish my study for a Master degree after a long time of continuous work. I am truly indebted and do appreciate many persons that have encouraged me through this hard yet challenging journey. Whilst being grateful to all of them, I must register my gratefulness to some particular individuals.

I would like to express my appreciation and deepest gratitude to my extraordinary supervisor Dr. Mazni Omar who advised, supported, and assisted me during all stages of this research. Without her ideas and comments, the accomplishment of my research would have been more difficult. She has always been there when I needed her invaluable feedback and everything related to accomplish this journey. It is an honor to be one of their students.

My deepest respect and thanks goes to my parents, my father Mohammed Dahr and my beloved mother Ameena Younis for their sacrifices, unconditional love and encouragement through their prayers. I value their endless efforts which made me who I am. A very special appreciation and sincere gratitude goes to my beloved wife (Israa Sabri) and my dear son (Ali) who were the biggest motivation for me to complete my study. Furthermore, I would like to thank my brothers and sisters for their encouragement, support and patience that keep me moving forward in completing this study as without them, I could not have completed this study.

I would like to extend my sincere thanks to my best friend (Nafea Raheem), who did not forget me during period of study and all time remind me. Further, words cannot express my gratitude to my friends, who always gave me the encouragement and help to complete this work. I will be forever thankful, grateful, and indebted to them

I express my deepest thanks to Ministry of Education in Iraq for helping and supporting me and giving necessary advices and guidance and arranged all facilities to make my study (Master Information and Technology) easier. Finally, special thanks goes to all staff of College of Arts and Science, University Utara Malaysia and those that contributed indirectly towards the success of my study.

**Thank you UUM**

## Table of Content

Permission to Use.....	ii
Abstrak .....	iii
Abstract .....	iv
Acknowledgement.....	v
Table of Content.....	vi
List of Tables.....	ix
List of Figures .....	x
List of Appendices .....	xi
<b>CHAPTER ONE: INTRODUCTION</b> .....	1
1.1 Introduction .....	1
1.2 Background of the Study.....	1
1.3 Problem Statement .....	3
1.4 Research Questions .....	6
1.5 Research Objectives .....	6
1.6 Significant of Research .....	7
1.7 Research Scope .....	8
1.8 Organization of the Thesis .....	8
1.9 Summary of Chapter .....	10
<b>CHAPTER TWO: LITERATURE REVIEW</b> .....	11
2.1 Introduction .....	11
2.2 Requirements Engineering .....	11
2.2.1 Requirements Development.....	13
2.2.1.1 Requirements Elicitation Phase .....	14
2.2.1.2 Requirements Analysis Phase .....	17
2.2.1.3 Requirements Specification Phase .....	18
2.2.1.4 Requirements Validation Phase .....	20
2.2.2 Requirements Management .....	21
2.2.2.1 Requirements Change Management .....	23
2.3 Requirements Traceability .....	24
2.3.1 Requirement Traceability Techniques .....	26
2.3.2 Requirement Traceability Tools .....	28
2.3.3 Factors in Selection of Requirement Traceability Practices .....	30
2.3.4 Related Works .....	33

2.4 Summary of Chapter .....	37
<b>CHAPTER THREE: RESEARCH METHODOLOGY .....</b>	<b>38</b>
3.1 Introduction .....	38
3.2 Research Procedure .....	38
3.3 Systematic Literature Review (SLR) .....	41
3.3.1 Formulating the Research Question .....	42
3.3.2 Constructing the Search.....	43
3.3.3 Study Selection.....	44
3.3.4 Data Extraction.....	45
3.3.5 Synthesis of the Extracted Data.....	45
3.4 Quantitative Approach .....	46
3.4.1 Populations and Sampling .....	46
3.4.2 Data collection Instrument.....	47
3.4.3 Data Analysis .....	49
3.4.4 Validation .....	50
3.4.5 Reliability Analysis.....	50
3.5 Qualitative Approach .....	51
3.5.1 Interview.....	52
3.5.2 Data Analysis.....	53
3.6 Summary of Chapter .....	54
<b>CHAPTER FOUR: RESULTS OF SYSTEMATIC LITERATURE REVIEW .....</b>	<b>55</b>
4.1 Introduction .....	55
4.2 Conducting the Review .....	55
4.3 Result of SLR.....	56
4.3.1 Requirement Traceability Techniques .....	58
4.3.2 Requirement Traceability Tools .....	69
4.3.3 Comparison between RT practices .....	73
4.4 Discussion .....	85
4.5 Summary of Chapter .....	89
<b>CHAPTER FIVE: RESULTS OF SURVEY AND INTERVIEW.....</b>	<b>91</b>
5.1 Introduction.....	91
5.2 Results of the Use of Traceability Practices .....	91
5.2.1 Demographic Characteristics of Respondents.....	91
5.2.2 Use of Requirements Traceability Tools .....	94
5.2.3 Use of Requirements Traceability Techniques .....	95



5.3 Reliability For Use of Traceability Practices .....	97
5.4 Descriptive Statistics .....	97
5.5 Analysis of Interviews.....	98
5.6 Discussion .....	103
5.7 Summary of Chapter .....	104
<b>CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS.....</b>	<b>105</b>
6.1 Introduction .....	105
6.2 Objectives Achievement .....	105
6.3 Contribution of the Research .....	107
6.4 Limitation of the Study .....	109
6.5 Conclusion and Future Works.....	109
REFERENCES.....	111



## List of Tables

Table 2.1: Requirements elicitation techniques. ....	15
Table 2.2: Specification techniques. ....	19
Table 2.3: Requirements management tools. ....	22
Table 2.4: Requirement traceability techniques.....	27
Table 2.5: Requirements traceability tools.....	29
Table 2.6: Summaries the prior studies related to RT practices .....	36
Table 3.1: The sections of questionnaire and the sources of the adapted questionnaires. .....	48
Table 3.2: Pilot test Cronbach's Alpha .....	51
Table 3.3: Differences between quantitative and qualitative approach .....	52
Table 4.1: Factors that help software companies to select the appropriate RT practices. .....	77
Table 4.2: Matching between the tools and techniques of traceability and the tools price .....	84
Table 5.1: Distribution of respondents.....	92
Table 5.2: Reliability Result .....	97
Table 5.3: Descriptive Statistics.....	98
Table 5.4: Participant's Information.....	99

## List of Figures

Figure 2.1: The Major Activities of Requirements Engineering.....	12
Figure 2.2: Requirements Elicitation Process .....	15
Figure 3.1: Proposed Research Framework .....	40
Figure 3.2: Systematic Review Guideline Process .....	41
Figure 4.1. Studies Selected of Digital Libraries .....	56
Figure 4.2: RT Techniques for Tracing Requirements .....	57
Figure 4.3: RT Tools for Tracing Requirements.....	58
Figure 5.1: Use of Traceability Tools in Software Companies.....	94
Figure 5.2: Use of Traceability Techniques in Software Companies .....	96



**UUM**  
Universiti Utara Malaysia

## List of Appendices

APPENDIX A: Questionnaire.....	134
APPENDIX B: Expert Reviewer.....	139
APPENDIX C: List of Software Companies .....	140
APPENDIX D: Interview Questions .....	141
APPENDIX E: Studies Selected of Systematic Literature Review .....	142



# **CHAPTER ONE**

## **INTRODUCTION**

### **1.1 Introduction**

This chapter starts by explaining the background of traceability. It then highlights the problem statement related to the study. The research objectives, research question, scope of the study, as well as the significance of the research, are also highlighted.

### **1.2 Background of the Study**

Lately, there is a rising interest in software systems which are capable of adapting to modifications in their requirements for the purpose of continuity in fulfilling their software's goals (Silva Souza, Lapouchnian, Robinson & Mylopoulos, 2011). According to Zainol and Mansoor (2011), requirements have a propensity to be modified during software development and these modifications must be managed. Also, according to the study's by Attarha and Modiri (2011), requirements are regarded as a challenging part in the software project because the requirements accurately determine what is to be produced. Requirements as defined by Zhou (2014), is a statement that recognizes a process or product's functional, operational, or design characteristic or limitations, which is unambiguous, measurable or testable, and mandatory for a process or product's acceptability by consumers or internal quality assurance guidelines. Similarly, Wen, Luo and Liang (2012) defined requirements as formal pronouncement of user's needs.

The root of requirements traceability is requirements engineering (RE). RE is considered as the initial phase of software engineering. According to Attarha and Modiri (2011), RE is a method for defining, modeling, recognizing, linking, documenting and maintaining or preserving software requirements in software life cycle

which gives a better understanding of the problem. In a nutshell, accomplishment of each software project is evaluated by putting the level of project's goals fulfillment into consideration. Hence, RE is a process of realizing these goals which is accomplished by identifying stakeholder and their needs, as well as documenting these needs in ways that responds to analysis, which communications and implementation of them in the future (Attarha & Modiri, 2011). However, the process of RE involves a large amount of data and inconstant requirements. Therefore, Requirements Management (RM) tools have been designed for the management of these changes (Zainol & Mansoor, 2011). RM manages modifications in the requirements all through RE procedure and development. Naz, Motla, Asghar, Abbas and Khatoon (2013) stated that RM involves activities associated with change identification, change maintenance, traceability and management of requirements changes (or change management). Furthermore, RM handles modifications to approve requirements while managing the relationship between requirements. In addition, dependency between requirement records or document, and other records produced all through the software engineering processes are also managed (Naz et al., 2013).

These types of problem can be addressed by a process of requirements management known as Traceability (Bashir & Qadir, 2006). Requirements Traceability (RT), is defined as (Gotel & Finkelstein, 1994):

*“the ability to follow the life of a requirement, in both a backward and forward direction” (p.1)*

Mader et al., (2013) defined RT practices as a set of steps or strategies prepared by project managers in the early stages of the project, and all information which must be documented using a traceability information model. Likewise, Cleland-Huang (2006) reported in their study that traceability practices consist of tools or methods that are

utilized in the implementation and maintenance of traceability links. Also, Katta and Stålhane (2014) elaborated that RT practices refer to tools and techniques used in tracing requirements during software development processes. In this study, RT practices refer to techniques and tools that are used in tracing requirements in the different stages of project development, to ensure that all requirements are fulfilled in the final project.

Traceability establishes a logical association between objects of the software development process (Gotel & Finkelstein, 1994). As an important process in modification management tasks, traceability provides relevant information on the potential penalties of a changing requirement (Mäder & Gotel, 2012). In spite of all these advantages, knowledge on the practical use of traceability in development projects is still lacking (Schwarz, Ebert, & Winter, 2010a). Given the merits of traceability requirement in company projects and products quality, this study focuses on RT practices in software companies in Malaysia. The major issues related to this phenomena are highlighted in the next section.

### **1.3 Problem Statement**

The requirements are the base of a software project development (Zaib, Chauhan, & Sirshar, 2015). Therefore, missing important requirements or capturing irrelevant requirements are the main causes of product failure (Khan et al., 2011). This is also supported by Khan, Khalid and ul Haq (2013), who asserted that the foremost reason of project failure is due to employing bad requirements practices. More precisely, according to Shubhamangala, Rao, Dakshinamurthy and Singh (2012), software projects usually fail due to poor requirements management. This issue is also discussed by Pandey and Suman (2012). In the same context, Marnewick (2014) stated that if a project begins with good requirements, success will be attained in terms of quality, cost and schedule. Thus, RM is very important or crucial for the success of any organization

or company. Rikhari and Kumar (2012) declared that poor requirement management leads to; poor quality software, rework, very low customer satisfaction, financial problem, delay in the delivery of the software, high cost, and low market value. All these problems result in software failure.

On the other aspect, traceability is one of the decisive activities of good requirements management. Traceability allows fast assessment of the impact when a change occurs (Ooi, Lim, & Lim, 2014). RT is an important part of development projects (Bouillon, Mäder, & Philippow, 2013; Mäder & Egyed, 2014). Therefore, researchers have shown that RT influences positively on the quality of software products (Cleland-huang et al., 2014; Spanoudakis & Zisman, 2005; Winkler & Pilgrim, 2010). In addition, Kirova, Kirby, Kothari and Childress (2008) explained that the direct benefits of traceability are an improved product quality, controlled requirements and effective response to changes. In the same vein, numerous studies also pointed out that the use of requirement traceability in a development project is frequently hindered by challenges in its execution and application, such as, insufficient tool support and ad-hoc traceability without strategy (Aizenbud-Reshef, Nolan, Rubin, & Shaham-Gafni, 2006; Gotel & Finkelstein, 1994; Ramesh, Stubbs, Powers, & Edwards, 1997; Schwarz, Ebert, & Winter, 2010b).

RT is considered as the heart of requirement management process. Despite the fact that RT is an area of much interest and concern in development projects (Mäder, Gotel, & Philippow, 2009a; Rempel, Mäder & Kuschke, 2013), surprisingly, it is rarely used (Ahmad & Ghazali, 2007; Arkley, Mason, & Riddle, 2002; Klimpke & Hildenbrand, 2009). Also, Regan, McCaffery, McDaid and Flood (2012) stated that most companies are either not implementing it, or they are lacking of guidance on how to implement effective traceability. In addition, there is a lack of empirical studies related to trace



requirements despite the availability of tools and techniques that support the tracing of requirement. Besides, prior studies still concentrated on the requirement traceability in general (Bouillon et al., 2013), where a few of the previous studies focused on stages of traceability, specifically in Malaysia. The aforementioned arguments encouraged the researcher to focus on RT.

Particularly, Bouillon et al., (2013) found out that the usage of RT is less common in development projects. The study by Solemon, Sahibuddin and Ghani (2010) shared this opinion and reported that less companies in Malaysia define traceability policies or maintain traceability manual in their projects. Saiedian, Kannenberg and Morozov (2013) pointed out that manual traceability methods are prone to errors and also time-consuming. They are also difficult to manage (Han, Youn, & Cho, 2014). Mäder and Egyed (2014) confirmed that, it is vital and essential to investigate whether utilization of RT can considerably support development activities.

According to (Cleland-Huang et al., 2011), for several research domains, open source projects may provide a wealth of useful data, this is not normally the case for traceability, and open source projects in this domain do not include any significant information about traceability. Therefore, identify factors for the selection of traceability practices will help companies assign the appropriate practice of their work.

In addition to the aforementioned arguments, Casey and Mc Caffery (2011) stated, there is a lack of guidance is available for practitioners to help them establish effective traceability in their projects. As a result of this, practitioners are ill-informed as to how best to accomplish this task, such as found in software companies. This opinion is supported by Regan, McCaffery, McDaid and Flood (2014). Notwithstanding the lack of guidance, there are a number of commonly accepted practices which can direct practitioners regarding the implementation and maintenance of traceability (Han et al.,

2014). In the same context, software companies' requirements are the greatest challenge to handle (Rosmadi, Ahmad, & Abdullah, 2015), due to constant modification. Therefore, this study aims at examining the practices of requirement traceability to software companies as well as to aid software companies in the selection of a suitable traceability practices for their work by identifying the RT practices factors. This study strives to assist software companies in addressing the lack of guidance on how to implement effective requirement traceability.

#### **1.4 Research Questions**

Based on the problem discussed in the previous section, this study aims at answering the following research questions:

1. What are the requirement traceability practices that have been used within software companies and factors that help these companies to select appropriate practices?
2. What are the current practices of requirement traceability that applied within software companies of Malaysia?
3. How to verify the requirement traceability practices used by the software companies?

#### **1.5 Research Objectives**

In order to minimise the gap in the literature review and also, to highlight of a suitable RT practices for software companies the following objectives are posed to achieve this purpose:

1. To identify RT practices within software companies and factors that help these companies to select appropriate practices using systematic literature review.

2. To investigate the current practices of requirement traceability applied among software companies in Malaysia quantitatively.
3. To verify the requirement traceability practices used qualitatively with industrial experts.

## **1.6 Significant of Research**

The significance of the study to requirement traceability is outlined below:

- i. This study seeks to enrich information about requirement traceability in literature, because there is a few studies focuses on this field.
- ii. As well as, in the systematic review section, the researcher also strive to shed light the various RT practices that are used to track requirements. Thus, this study will motivate the new researchers to further research based on these issues.
- iii. Shedding light on factors that facilitates the selection of RT practices by software companies.
- iv. There is paucity of the empirical study on the software companies related on the requirement traceability, therefore this study highlight the current practices used by software companies in northern region of Malaysia.
- v. The study also sheds light on the reasons that make software companies less attention in using RT practices in software development processes.
- vi. Lastly, in Malaysia particular, there is still lack of research on requirements traceability practices among software companies.

In a nutshell, the outcomes of this study is to fulfill the gap of the lack of the empirical evidence related to requirements traceability practices used by the software companies in Malaysia.

## **1.7 Research Scope**

RT is a part of the requirements management that used to increase the quality of the product during the software development life cycle (SDLC). With regards to that, this study looks into the requirements traceability and identify practices used to it. In terms of the users, Sekaran and Bougie (2010) stated, in research involving a wide range of elements, it is practically impossible collect data from, or test or examination of each element. Even if it were possible it would be high-priced in terms of cost and need a long time as well as other human resources. Therefore, data was gathered from practitioners in software companies in various parts of the northern of Malaysia and previous studies conducted on the requirements traceability in literature. In addition, the northern region was selected because of the availability of software companies that fit into this study.

This study used a snowball sampling method for determining the sample of study because of the desired sample characteristic is rare as well as the number of software companies in northern region is not constant. Snowball sampling is a special non-probability method used when the desired sample characteristic is rare (Aartsengel & Kurtoglu, 2013). In snowball method, the researcher asks participants to identify others to become members of the sample (Creswell, 2012). This method is suitable for the study for the following reasons; identify participants in the study by people with experience in this area, reduce the cost and time required for data collection, and this method is optimal to get to the hidden populations.

## **1.8 Organization of the Thesis**

This thesis is organized in six chapters. The following is an outline of the main contents of each chapter:

## **i. Chapter 1: Introduction**

This chapter explains the overview of requirements traceability as well as an outline of the most important studies in this domain. Based on these studies, the problem statement had been formulated, research questions, objectives and scope of the study. The last section of the chapter explains the significance of the study.

## **ii. Chapter 2: Literature Review**

This chapter reviews the most important studies that are relevant to requirements engineering and its branches. Meanwhile, this chapter also covers the information sharing studies related to requirements traceability as well as the techniques and tools that are used to track requirements. Also, it includes challenges that hinder the implementation of traceability and related works in this area.

## **iii. Chapter 3: Research Methodology**

Chapter 3 focuses on the methods used to solve the identified problem and achieve the objectives of this study. It includes details and justifications about research methodology used as well as explain steps of data collection and analysis.

## **iv. Chapter 4: Results of Systematic Literature Review**

This chapter presents the findings and analysis of systematic literature review about requirement traceability practices. Additionally, factors of RT practices extracted from studies that are enable companies to choose the appropriate

practice of company's work. In the end, discuss the results of a systematic literature review

## **v. Chapter 5: Results of Survey And Interview**

Chapter 5 presents the results of a questionnaire on use RT practices in software companies as well as results of interviews about reasons that are prevent software companies from use RT practices. the final section of this chapter discusses the results of questionnaire and interviews.

## **vi. Chapter 6: Conclusion and Recommendations**

Finally, the thesis ends with a discussion of the objectives achievement, limitations of the study and contribution as well as conclusion and future works.

### **1.9 Summary of Chapter**

This chapter illustrates the importance of RT of the software, and the motivation for this study. As well as, the importance of requirements engineering and requirements management. This will serve as an introduction into the topic of discussion. The problem statement of this study is also clearly elaborated, coupled with the research questions of this study and its objectives. In addition, the significance of the study for researchers and practitioners and research scope.

## **CHAPTER TWO**

### **LITERATURE REVIEW**

#### **2.1 Introduction**

This chapter presents a set of studies and research that are relevant to this study. It starts with requirements engineering which is the root of the requirements traceability, and its branches. In addition, the chapter discusses vividly the importance of requirement traceability in more detail, challenges impeding the implementation of traceability and previous works that are related to the research topic.

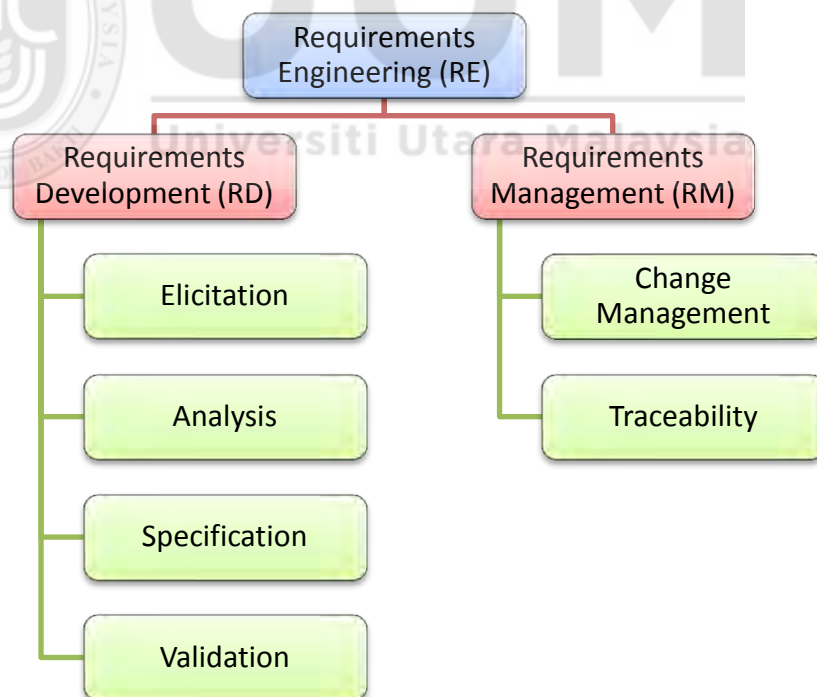
#### **2.2 Requirements Engineering**

Requirement Engineering (RE) as a field, plays an important part over the entire process of product development (Pandey, Suman, & Ramani, 2010). The method of integrating RE in the product development cycle is largely affected by the acceptance and performance of a product in a market (Anitha & Prabhu, 2012). RE is considered as the most significant section of software engineering and perhaps of the whole software life cycle. This is because, mistakes made at this stage, can be very costly if doesn't manifest till a later stage of the software development. Otherwise, if the requirement specifications are identified correctly, there will be minimum error at the later stage (Chakraborty, Baowaly, Arefin, & Bahar, 2012).

RE is the initial and decisive phase of software engineering. According to Mohebzada, Ruhe and Eberlein (2012) elaborated that the specification of requirements evolves over time, reflecting the realities of the project. Hence, it is not a one-time task. This creates a more challenging task, as incomplete or untrue, leading to the main causes of the project failures. Companies differ in their need for requirements engineering. Haron et

al., (2012) stated that, the success of a software project is due to the application of RE in their development process. RE is a name for a structured set of activity that helps to develop the understanding and documentation of system specification for IT personnel's and stakeholders involved in the system development. In the same vein, the techniques are used in gathering to develop a system which performs tasks usually conducted by human experts.

RE is the discipline of analysing, determining, documenting, pruning, and validating the needs and requirements of stakeholders at specific system (Agarwal & Gael, 2014). Jiang et al., (2008) stated that project failures are contributed by bad requirement engineering. Therefore, appropriate techniques should be utilized, preventing wastage of time and cost, thus, avoiding a redo of the system development. Fig. 2.1 illustrates the main activities of requirements engineering.



*Figure 2.1: The Major Activities of Requirements Engineering (Soonsongtanee & Limpiyakorn, 2010; Wiegers, 2000)*



Requirement engineering comprises of two main types of activities. The first is requirements development and the second is requirements management. For requirements development, it includes four activities are elicitation, analysis, specification and validation. Requirements management has two activities are change management and traceability. The next sub-section will discuss in details related RE activities and requirements development.

### **2.2.1 Requirements Development**

Requirements development (RD) is an activity of comprehending, eliciting and analysing clients' or customers' expectations (Zhang, 2007). During this phase, the development team sits alongside the product owners, in order to identify their needs or requests by listing the features required for the system. The team gathers much needed information from the product owners for the purpose of understanding their needs. The information gathered will be further analysed to acquire the user requirement (Sarkan, Ahmad, & Bakar, 2011). In the same manner, Diev (2007) stated that, requirements development as a design activity involves breaking the system into sub-systems, and establishing how these subsystems should interact. It also involves specifying requirements for subsystems.

Khan et al., (2013) pointed out that, every requirement has a life span starting from its origin to implementation. A single requirement is also capable of affecting other requirements in a positive or negative manner. Requirements development passes through the following phases: requirements elicitation phase, requirements analysis phase, requirements specification phase and requirements validation phase (Khan et al., 2013). The following section will discuss the requirements elicitation , which is the first stage of the requirements development.

### **2.2.1.1 Requirements Elicitation Phase**

Requirements elicitation phase is collecting information properly, to achieve it, must have a clear understanding of the requirements and determine its value for software. In addition, Yozgyur (2014) defined the requirements elicitation as a stage of RE where the required information is extracted and collected from the stakeholders like customers, users etc. Sutcliffe and Sawyer (2013) stated, elicitation still remains problematic; missing or mistaken requirements still delay projects and cause cost overruns.

In addition, Sharma and Pandey (2014) found out that, various errors related to requirements are avoided to the later or final phases of the development life cycle. Correcting these errors after or during the execution of the software, increase efforts and cost unnecessarily. This point strengthens the fact that additional attention should be given to requirements elicitation since receiving imprecise requirements from users can result in wrong designing or may be difficult to resolve at the later stage.

According to Sadiq and Jain (2014), a successful software system is a system that completes on time, within budget and meets customer requirements. In addition, the authors have identified some of the primary causes that lead to the software failure, such as lack of user involvement, incomplete requirements and lack of executive management support. Moreover, they stated that it can't elicit a full set of requirements unless it's determined by the stakeholders in the requirements elicitation process.

Burnay, Jureta and Faulkner (2015) stated that, the lack of eliciting completeness of the information is common and has been recognized as a major challenge in RE. Fig. 2.2 illustrates the phases of requirements elicitation process.

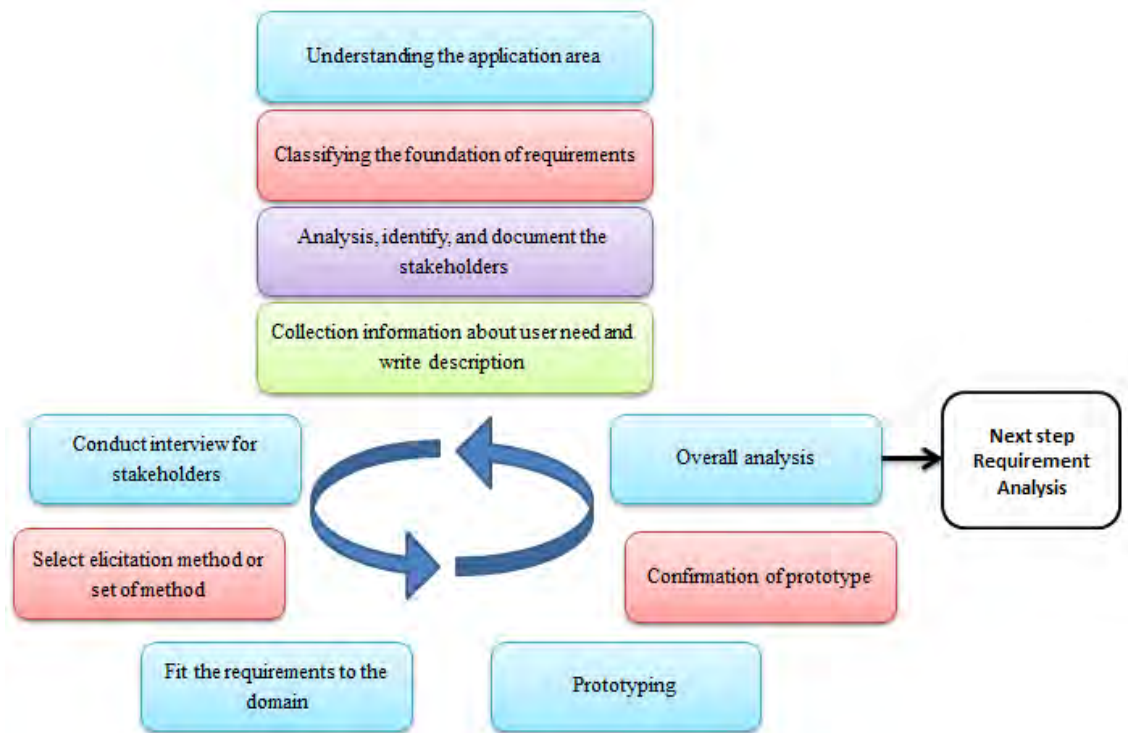


Figure 2.2: Requirements Elicitation Process (Sharma & Pandey, 2014)

There are many techniques used to elicit information from stakeholders or the user. Table 2.1 shows a set of elicitation techniques.

Table 2.1

*Requirements elicitation techniques (Fricker, Grau, & Zwingli, 2015).*

Technique	Description
<b>Archaeology</b>	Analysis of current systems to comprehend their quality, functionality, and usage.
<b>Creativity</b>	The selection and generation of concepts to innovate or solve a challenging problem.
<b>Data mining</b>	Searching and filtering of requirements databases to detect significant knowledge about the needs of stakeholder.

---

<b>Interview</b>	Meetings between stakeholders and requirements engineers to discuss important issues for the system.
<b>Introspection</b>	Use of domain knowledge merged with empathy and reflection to base requirements on experience.
<b>Observation</b>	Study of use of system, by real users and probably in the target environment, to comprehend processes, usage, strengths, and weaknesses of the existing system.
<b>Questionnaire based survey</b>	Electronic or paper form with questions and space or options for feedbacks, distributed to stakeholders to acquire stakeholder opinion through an overview.
<b>Reuse</b>	Use of existing or current specifications to evade reinvention of requirements that are adequate already.
<b>Workshop</b>	Meeting between stakeholders and a requirements engineer to come to an agreement between the workshop participants.

---

Elicitation is the first phase of the requirements development. According to Sharma and Pandey (2014), the major aim of this stage is to categorize the system limits and specify the communicative and efficient characteristics of a system and the accomplishment of this process according to the recognition of the appropriate stakeholders from various background and identifying their requirements. In general, requirements elicitation approaches or techniques that can't possibly be suitable for every project. The choice of approaches to be includes is based on the particular environment of the project and chosen of the techniques based on analyst perspective or determined by a particular methodology. Requirements analysis phase starts when this phase is completed.

### **2.2.1.2 Requirements Analysis Phase**

Requirement analysis phase involves refining the requirements to confirm that all the stakeholders apprehend them, and also inspecting them for errors, omissions or oversights, and other deficiencies. Furthermore, requirement analysis involves disintegrating high-level requirements into the right level of details, constructing prototypes, evaluating feasibility, and negotiating priorities (Wiegers & Beatty, 2013). Requirements that do not meet the perspective of stakeholders may have caused the failure of the software. This is shown by Chaos (2014), who confirmed that, data from the Chaos Report of the Standish Group indicates that the failure rate of software projects were two in three projects. One of the major causes of this is unsatisfied stakeholders and customers. Hence, the analysis of those requirements becomes important, where it should be taken into account, that these requirements meet most of the views of stakeholders, whereas, these requirements can be completed without exceeded the time limit for the software as well as taking into account the cost of the project.

Most studies on requirements analysis concentrate on novel or improved procedures for evaluating the quality of documented requirements. A number of analyses search for well-formedness errors in the requirements, where an error can be inconsistency, incompleteness, or ambiguity. Other analyses search for anomalies, such as unidentified interactions among requirements, potential obstacles to the satisfaction of requirements, or missing assumptions. Both categories of analyses disclose misunderstandings or questions pertaining to the requirements that usually calls for further elicitation. Requirements analysis also includes procedures, such as risk and impact analysis, which gives specifiers a better understanding of the requirements, their potential consequences, and their interrelationships, enabling the specifiers to make more-informed decisions

(Cheng & Atlee, 2007). In fact, requirements analysis is not merely a specification of the system, but is a private specifications of the final product. The following section explains the third phase of the requirements development which is the requirements specification phase.

### **2.2.1.3 Requirements Specification Phase**

Requirements specification phase which is sometimes called software requirements specification or system requirements specification (SRS), is a document that explains multiple technical issues of a software system (da Silva, 2014; Elliott Sr & Allen, 2013). In addition, Juergens et al., (2010) explained that SRS are the basis of many software development projects. They have a strong influence on the quality of the developed product and also on the effort put in development, which is as a result of their pivotal role in the software development process. Besides, they are normally the central, and often only, communication artefact utilized between customers and contractors. Therefore, the quality of SRS is of utmost importance to the achievement of a successful software development project. The specification process still includes significant human-centered efforts, consisting of inconsistency-prone and notably error. Varying phases of the process, may contain error; from the requirements specification phase, to the design, actual specification, and the implementation of the software models.

When inconsistencies or specification errors are detected, current tools provide partial information about the rectification to be applied on the system being developed (Borges, Garcez, Lamb, & Nuseibeh, 2011). The most important techniques that are used for the specify of requirements that have been analysed in the previous phase are shown in Table 2.2.

Table 2.2

*Specification techniques (Fricker et al., 2015).*

<b>Technique</b>	<b>Description</b>
<b>Natural language</b>	Specifying requirement or needs with words/sentences to obtain flexibility, specification, and understandability. The use of language templates can be applied to enhance precision.
<b>Structured Analysis Diagrams</b>	Specifying functions, structure, processes, and behavior with one of the graphical notations projected by structured analysis to make structure visible and obtain precision.
<b>Tables</b>	Specifying concepts to comprehend the rules or terminology for how conditions influence the behavior of a system.
<b>UML Diagrams</b>	Specifying functions, processes, scenarios, relations, rules, behavior, and deployment with graphical notations from the Unified Modeling Language to show structure and advance precision.
<b>User screens</b>	Specifying the user interface with paper or tool-based mock-ups increase the tangible and the authenticity of the planned system.

According to Fatwanto (2013), it is common for requirements are specified by using natural language (NL), that to be easy to understanding between stakeholders or potential users and developers due to natural language flexibility and simplicity, and must both parties have to share their internal view regarding the system under consideration in order to obtain a common understanding.

Furthermore, Sharma, Bhatia and Biswas (2014) confirmed that, NL is the most popular and the most desired forms of requirements expression for an envisioned software system. NL is a suitable choice for requirement specification documenting, because it is comprehensible by all the stakeholders participating in software development, and is also quite expressive. The final phase of the development requirements will be discussed in the next section.

#### **2.2.1.4 Requirements Validation Phase**

Requirement Validation Phase is a method of determining if the specification is a precise representation of the customers' need. Validation responds to the question, "Am I building the right product?" (Laplante, 2013). According to Raja (2009), the process of requirement validation confirms that, the requirements in SRS are complete and consistent. The process attempts to detect the errors in SRS, prior to being used as a foundation for further system development. Furthermore, Yousuf, Zaman and Ikram (2008) explained that the aim of requirements validation is not to ascertain the correctness of the requirement, but to detect and correct all errors (inconsistencies, incorrect information, and omissions).

Furthermore, Felderer and Beer (2013) stated that, great significance can be achieved from the quality of requirements for the software development lifecycle, since it affects every step in software development. Validation techniques are practiced to assure completeness of requirements and their quality features such as traceability or verifiability and comprehensibility. These are the main challenges in software development lifecycle. Requirements validation approaches are one of the most efficient approaches for guaranteeing a successful software development. There are many techniques that used to verify the requirements to ensure there are no errors, ambiguities and so on. The most commonly used techniques are prototypes, animation,



inspection, review, natural language paraphrasing and expert system approaches (Yousuf et al., 2008). The second major phase of requirements engineering is requirements management that will be discussed in the next section.

### **2.2.2 Requirements Management**

Requirements Management represent a description what should be done by the system (Ferreira & da Silva, 2008). At the beginning of any software development process, information will be collected depending on the customer's need or user, and this information is analysed to extract requirements. According to Alghazzawi, Siddiqui, Bokhari and Hamatta (2014), the whole requirement being analysed should be managed appropriately for a successful project to be achieved.

The management of multiple thousands of requirements, can be a tough task that requires more than just a spreadsheet. Requirement management executes a vital role in the achievement of a successful software project, as a business depends very much on software for efficient conduct and mission-critical functions.

Requirements management includes those tasks that documents and maintain the progressing requirements, associated context, and historical information, from the requirements engineering activities. Furthermore, RM also establishes techniques for defining, controlling, and publishing baseline requirements used by all levels of the system-of-interest (Stallinger, Neumann, Schossleitner, & Zeilinger, 2011).

The practice of managing requirement is a challenging task, especially when the amount of requirements are in thousands and the changes are many. RM tools cannot increase the quality of requirement gathering. Rather, it eases practice implementation and increases their efficiencies for project managers and business analysts. Over the past few decades, a number of RM tools have been developed and made available in the

market for varying projects. The tools, manage the relationship between business requirement, functional requirement and technical specification. Many tools are used in requirements management, such as Rational RequisitePro, CaliberRM, and DOORS (Alghazzawi et al., 2014). Tools that are used to manage requirements is many and varied. Table 2.3 shows the most important tools used in this area.

Table 2.3

*Requirements management tools (Shahid, Ibrahim, & Mahrin, 2011).*

<b>Tool</b>	<b>Description</b>
<b>DOORS</b>	Doors are a RM tool designed by Telelogic. After that IBM acquired in 2008. It provides easy and flexible method to generate and traverse the associations between several requirements.
<b>Rational RequisitePro</b>	It is also a tool for managing requirements produced by IBM. It supports manage of requirements by project teams to attain better traceability, write decent use cases, decrease project redo, and promote collaboration as well as quality.
<b>XTie-RT</b>	It is a potent RM and analysis tool used in the management of critical programs.
<b>CaliberRM</b>	It is a tool designed by Borland for RM. It captures and manages business, functional, technical, and operational requirements.
<b>TopTeam Analyst</b>	It is a thorough solution for gathering and management of requirements. It supports use cases and test cases to preserve the SRS process.

---

**ReMa**

This is an acronym for requirements manager. It is a systematic and significant tool, that assists project managers in tracking requirements, as well as managing them through the entire software life cycle.

---

There are many RM tools which are obtainable in literatures and on the Internet, commercially or the laboratory version. Each tool is designed for a particular purpose and within a particular field. For example, Shahid et al., (2011) explains that, DOORS helps to decrease costs, increase efficiency, and improve quality by enabling the optimization of requirements communication, collaboration and verification. Shahid et al., (2011) also explains that CaliberRM allows stakeholders across the organization to cooperate effectively, promoting timely delivery of project, and within budget and specification as well. The first part of the requirements management that will be discussed in the next section is requirements change management.

#### **2.2.2.1 Requirements Change Management**

Recently, software systems get more and more complex. Following the evolution of the business needs, the software system requirements change constantly and new requirements arise frequently. The addition of the new and/ or reformed requirements with the existing ones, comes with the need for adapting the architecture and source code(s) of the software system and to satisfy the new sets of requirements. The incorporation of the new/modified requirements and the adaptations to the software system, is known as change management. The complexity and size of the software systems make change management time consuming and costly (Goknil, Kurtev, van den Berg, & Spijkerman, 2014).

According to Khan, Khalid and ul Haq (2013), changes to requirements are very critical steps in any of the software development process. Requirements change management has been an open issue in the literature for several decades and numerous requirements change methods have been suggested, but none completely copes with it.

Requirements changing has been considered a challenging area of research or study. It has been noted that requirements changes during various phases of the SDLC plays an important role in the success or failure of any project. As a matter of fact, over half of the system's requirements will eventually be modified before the actual system deployment. Many reasons can be related to requirement change such as user preferences, ever changing environment, and change in technology. These causes can arise during any phase of SDLC which in turn, leads to changes in the system's overall requirements (Bano, Imtiaz, Ikram, Niazi, & Usman, 2012; Manapian & Prompoon, 2014).

Kang, Kim, Kang and Eom (2014) explained that traceability allows the developer to keep track of changes or modifications in software development projects. In line with this, Kirova et al., (2008) pointed out that requirements changes are unavoidable during the product development. Thus, RT practices can be used in order to track the requirements changes. the next section will discuss in details about requirement traceability.

### **2.3 Requirements Traceability**

Most of the disciplines in requirements management focus on traceability, which is the ability to trace requirements as they are converted from customer needs to design specifications, via the layers of design. Efficient tracing relies on requirements being expressed as singular, concise, and unambiguous statements suitable to their level of

abstraction (Dick, 2012). According to Kang et al., (2014), traceability is an association between the product features, and the set of product requirements. It is denoted as a set of mappings where each mapping possesses a characteristic of the source element, and also a set of requirements for the target element.

The importance of traceability has been widely recognized, and it is a practice prescribed in many development standards (Nair, De La Vara, & Sen, 2013). In the same vein, RT is very important in the RM, where Zhou (2014) pointed out that, RT is the heart of the requirements management. Lago, Muccini and van Vliet (2009) stated, RT (or traceability for short) is a significant mechanism for the managing and auditing of the entire software development process. Furthermore, Kong and Yuan (2009) stated, RT is considered to be the key factor of a highly effective software engineering management, and an improved quality of software system. Also, Ramesh and Jarke (2001) stated that, traceability provides visibility into the required aspects of software and systems development process. This gives a better comprehension of the software system under development.

In the same context, Kirova et al., (2008) explained that the direct benefits of traceability to the product organization as well as the end customers, are an improved product quality, controlled requirements, effective response to changes, and product churn, which is as a result of minimizing the product implementation interval. Many agencies, customers, standards, and quality frameworks, need requirements traceability, while a few define how the implementation of the traceability. This leads to an extensive range of variability in the benefits and implementations of traceability. The benefits can be abundant when a product organization utilizes a comprehensive requirements traceability process which is tailored to its needs and characteristics.

Furthermore, Gotel and Finkelstein (1994) discovered that, traceability failures primarily happen due to a breakdown of communication among developers, inadequate scheduling time, shortage of support tools, and an opinion from the product team that the effort needed in maintaining traceability was excessive. The cost of implementation against the benefit of effective traceability can be enhanced by choosing an appropriate implementation strategy for the product team. In addition, Nair et al., (2013) found, a few studies on traceability practices compared with the importance of traceability in software development. In this study, RT practices refers to RT techniques and tools used to track requirements. The next sub-sections will discuss further of RT techniques and tools.

### **2.3.1 Requirement Traceability Techniques**

Traceability relates the requirements of a system, and system comprehension efforts are minimized with the help of its source code. It is also necessary to ensure that a system's source code always suits its requirements and that only the identified requirements have been implemented by the developers. Yet, in the course of software evolution and maintenance, as developers remove, add, or modify features in general, requirement traceability links become outdated because developers do not or cannot put in an effort to update them. Hence, recovering these traceability links afterwards, is a daunting and expensive task for developers. Therefore, techniques have been developed for the purpose of recovering these links semi-automatically or automatically (Ali, Gueneuc, & Antoniol, 2013). Table 2.4 shows a set of the techniques used in requirements traceability.

Table 2.4

*Requirement traceability techniques*

<b>Technique</b>	<b>Description</b>
<b>Information Retrieval (IR)</b>	IR is a technique used to create traceability link between requirements documents and source code, and developers can be automatically advised which source code documents to edit to address (Thomas, Adams, Hassan, & Blostein, 2014).
<b>Event-based (EB)</b>	EB is a technique used for updating and preserving traceability relationships. When change occurs in requirements, the dependent artifacts are notified and subsequently proper action can be taken (Rochimah, Wan Kadir, & Abdullah, 2007).
<b>Rule-based (RB)</b>	Rule-based is one of the wide-spread techniques for maintenance of post-requirements traceability relations. RB is a technique which anchors the automatic maintenance of traceability links between analysis, requirements, and design models of software systems. This technique is based on rules which permit recognition of development tasks within a flow of elementary modification events. These rules make a directive available for updating traceability relations or links in predefined manners (Mäder, Gotel, & Philippow, 2008).
<b>Value-based (VB)</b>	This approach offers technical support to carry out

---

requirements tracing in addition to taking cost and value considerations into account. As a result, it delivers an economic model and a technical model for requirement tracing based on certain criteria (Heindl & Biffel, 2005).

**Scenario-based (SB)**

SB technique utilizes a hypothesized trace information which must be entered manually. Then, it create trace links using runtime information. Test case scenarios are implemented on a running system and implementation information is acquired with the help of a monitoring tool. This information is merged with the hypothesized trace information, resulting in a footprint graph. The relationship between artifacts in the system is displayed on the graph. The traceability links are automatically created, nonetheless, the hypothesized trace information must be entered manually (Rochimah et al., 2007).



---

Table 2.4 illustrates several techniques used to trace RT during software development. There is a difference in the use of traceability techniques in terms of the type of requirements that will be tracked. Some of techniques are used to follow the functional requirements, while there are techniques used to follow the non-functional requirements, and there are techniques for the purpose of tracing both types as well.

### **2.3.2 Requirement Traceability Tools**

Common requirements for the management of tools address traceability problems by offering support links between levels of requirements and permitting the storage of pointers to other artefacts that are related to requirements (normally as attributes to the



requirements). In a single-vendor environment, incorporation with other tools is sometimes available, adding extra flexibility to the whole traceability solution. In a mixed environment, components for integration are often developed for the importation of traceability information into the requirements management tool. This is a cost-effective and viable solution for organizations and projects where access to the requirements management tools, is always available to every participant in the development lifespan, and where forward traceability and navigation are more significant, and the constrictions and predefined traceability semantics that come along with the tools, are appropriate for the projects' needs. In addition, if the tools are open and accessible, they can function as the basis for the development of an advanced and even more flexible traceability environments most especially for mid-size organizations and projects (Kirova et al., 2008; Ramesh et al., 1997). Table 2.5 illustrate some of the tools used in the trace of requirements.

Table 2.5  
*Requirements traceability tools (Mäder et al., 2008; Shahid et al., 2011)*

Tool	Description
<b>DesignTrack</b>	DesignTrack is a prototype tool for RT support. It offers traceability between architectural design and requirements.
<b>RETRO</b>	Requirements tracing on-target (RETRO) is an tool for RT that facilitates the automated generation of RT matrices. It uses an information retrieval (IR) approach and possesses a GUI front-end.
<b>DevComplete</b>	DevComplete is an RT tool designed by SmartBear software. It

---

offers full or complete traceability among project tasks, requirements tracking, and defects to improve team agility.

**traceMAINTAINER** This is a tool for RT which preserves post-requirements traceability amongst the features of structural UML models.

---

Some of these tools are designed for the purpose of traceability, whereas other tools created to manage of requirements but also support tracing of requirements. For example, Shahid et al., (2011) stated that, Design Track is a tool used in RT to provide a navigation environment for complex or multiplex design information spaces through the supporting of requirement traceability. In addition, it assesses the drawbacks, power, and applicability of the requirement traceability enabled computer-aided design.

### **2.3.3 Factors in Selection of Requirement Traceability Practices**

Literatures have identified the merits of requirement traceability in requirements management. Zhou (2014) stated that, requirement traceability is considered the heart of the requirement management process. RT does not only offer a significant support for system developers through the entire SDLC, it is also an essential requirement for the certification process. Requirements tracing can be of benefit in executing change impact analysis, criticality analysis, risk analysis, and regression testing; though, it is not limited to any of these (Zhou, 2014). Furthermore, traceability assists software engineers in the effective development and management of software systems (Hassnain, 2015). Lack of traceability results in budget overruns and project failures, which leads to poor software maintenance.

Practitioners face numerous challenges when implementing effective traceability in their projects. Zhou (2014) pointed out that the issues consist of technical challenges

associated with physically creating, maintaining, recovering, and utilizing thousands of relatively vulnerable and interrelated traceability links. Again, Cleland-Huang, Chang and Christensen (2003) explained in their study that the various challenges faced while implementing traceability are the cost and time for tracing requirements, insufficient resources, lack of RT practices training, and lack of coordination between people which were responsible for different traceable artifacts.

The main challenges that facing traceability are financial issues and, the lack of knowledge or prevalence traceability practices. Jaber, Sharif and Liu (2013) reported that additional training on the use of traceability tools has helped in overcoming some challenges in software development such as time and effort, but this of course, requires much funding. In the same context, Winkler and Pilgrim (2010) elaborated that the core challenge in RT is the economics behind implementing traceability practices. If this factor is overlooked, it is obvious that the best traceability could be attained by taking record of any trace to the highest extent possible.

Furthermore, there is insufficient knowledge or the prevalence of practices that are implemented in trace requirements. Despite the availability of numerous techniques and tools that support the tracing of requirement, there is still insufficient empirical studies showing if and how the practice is put in actualization in the development process (Blaauboer, Sikkel, & Aydin, 2007). More so, a greater number of reports are being published by practitioners, covering wider experience or information on traceability for use by the wider traceability community. Though, the problem of confidentiality which restricts progress still remains. When organizations perform traceability methods and techniques which does not operate as intended, the results are not always published, posing a challenge for the traceability community to learn what does and does not work or operate with time (Gotel et al., 2012). The study by Cleland-Huang et al., (2011)

shows that a primary limitation to benchmarking and standardization in the traceability community, is the issue of obtaining non-trivial datasets. Although industries occasionally make data available or grant limited data access to individual research groups, sharing is always done under a non-disclosure agreement. Consequently, the practitioners have no knowledge of the techniques and tools used in implementing effective traceability in development projects. Thus, more research related to trace requirements in the domain of software industry is required to ensure increase percentage of the project's success and increase product quality as well.

There are a range of factors that can help in the selection of RT practices such as: identify practices that operate automated, semi-automatic and manual, identify type of requirements that are tracked by these practices, as well as identify the processes taking place on traceability links like creation, maintenance and updating these links, and size of projects. Ali, Sharafi, Guéhéneuc, and Antoniol (2015) stated, the process of create traceability links is a laborious task and take more time and effort by developers during development project. In addition, Blaauboer (2006) pointed out, most of the changes in requirements occur during software maintenance process, especially when using an iterative development methodology. Therefore, identifying RT practices used to create, maintain and update traceability links in addition to the method of work these practices can help in the selection of practice required by the software companies.

According to Cleland-Huang et al., (2011), complex systems are most in need of traceability to ensure the success of the project by tracing requirements and artifacts, especially large systems. Moreover, Requirement traceability is an important and critical activity, especially for medium and large software projects (Cleland-Huang & Schmelzer, 2003). It is very important to trace the functional and non-functional requirements, to ensure that the project meets all the views of the stakeholders.

Therefore, Identify practices that are used to trace the functional and non-functional requirements as well as the size of the project, it will help to choose the appropriate practices. In the following section, the related studies about requirements traceability practice will be discussed.

#### **2.3.4 Related Works**

During previous studies of RT practices, authors have pointed out that the practice is a tool used to trace requirements and used to determine if the system code cover all system requirements. Whereas, others said that practice is a technique used to ensure that all the requirements that have been extracted, analyzed, and verified that have been included in the software and there is no ambiguity or loss of those requirements (Katta & Stålhane, 2014; Winkler & Pilgrim, 2010).

In this study, requirements traceability practice refer tools and techniques that used for tracing requirement during software development processes. Gotel and Finkelstein (1994) describe the results of empirical studies into traceability, that was conducted in 1992 and lasted throughout one year. About one hundred software development practitioners with approximately 30 years of experience holding diverse position, in a large organization, were involved in the study. Five focus group sessions were held and attended by thirty-seven (37) practitioners for the consolidation of data and for independent observation of the process of gathering requirements. Also, a comprehensive questionnaire was issued, and development workshops were held. They found multiple perspectives on the expectations from traceability, problems encountered, and conflicts evident between parties in charge of producing traceability and the users of traceability. Furthermore, the authors found that the pre-requirements traceability (pre-requirement traceability refers to the source and origin of a requirements, while post-requirement traceability refers to those aspects of a

requirement's life that result from inclusion in the requirement specification.) need more attention, so the need to merge a wider set of data in traceability, such as the people involved in the project and source material.

Research by Ramesh and Jarke (2001) was on a large practitioner study of traceability, which took over three years for data collection in the 1990's. Fifty-eight (58) students of masters in information technology were included in a pilot study to produce an opening traceability meta-model and document the scheme and achievements of the work. Thirty (30) focus group discussions consisting of five (5) people (each from twenty six companies) made up the main study. Their main emphasis was the kind of traceability link that was utilized in ideal and current practices. The results indicated that traceability links must be strong and clear to avoid semantic misinterpretation. As a result, the authors proposed for practitioners a traceability reference models and meta-model to assist them in traceability. The use of a traceability information model as a necessary condition to employ traceability is advocated.

According to Ahmad and Ghazali (2007) study, their study was conducted on fifteen practitioners and three IT companies. The practitioners, who develop small projects with practical experience of 6 – 10 years, were interviewed. Also, the project documentation of the companies was analysed. The result of their findings is that subjects viewed pre-requirements traceability as being more useful than post-requirements traceability. The researchers failed to expose the importance or advantages of traceability to their topics in details.

Mäder and Egyed (2012) performed a controlled experiment consisting of fifty-two (52) subjects conducting three hundred and fifteen actual maintenance jobs on two third (2/3) party development projects. Half of them with and the other half without traceability. Their discovery was that the performance of those with traceability was twenty one

percent (21%) quicker with tasks and generated sixty percent (60%) more accurate solutions. The result of traceability corresponds to the type of task performed. Particularly, the gain in correctness differed strongly between tasks. Furthermore, indications from the data shows that the gain of correctness via traceability is minimized after performance of the first activity or task. Subjects that did not use traceability produce more accurate results next to the main task, while subjects that utilized traceability performed excellently right from the start. The researchers target was an initial cost-benefit estimation, and setting the time reductions measured by using traceability in as related with the initial costs of setting-up traceability for the evaluated systems.

The study by Bouillon, Mäder and Philippow (2013) described the design of a survey to gather information on the use of requirements traceability by practitioners in development projects. The researchers collected 29 constantly cited usage scenarios of traceability based on a literature study. The study reported that 56 participants that applied practical requirements traceability used 42% of the 29 scenarios constantly. The operations of every scenario were analysed and the result was that requirements management and engineering, compliance demonstration, and project management are the domains in which traceability is mostly applied. The study also discovered that the use of traceability during design, implementation, evolution, and software maintenance is not common. In a nutshell, practitioners account their challenges with bad cost-benefit ratio, using their traceability. The solution to this difficulty is by using additional integrated tool support and method. Table 2.5 show in briefly studies that are related on RT in practices.

Table 2.6

*Summaries the prior studies related to RT practices*

No	Authors	Year	Finding	Limitation
1	Gotel and Finkelstein	1994	Multiple perspectives were on the expectations from traceability, problems encountered, and conflicts evident between parties in charge of establishing traceability and those using it.	There was no report to their subjects on the actual benefits of traceability.
2	Ramesh and Jarke	2001	An agreement on a traceability meta-model, and also on establishing reference models for other practitioners' use.	There was no exposure or documentation of the genuine benefits of traceability to the subjects.
3	Ahmad and Ghazali	2007	The result of the findings was that subjects consider pre-requirements traceability to be more useful than post-requirements traceability.	The researchers did not examine the benefits of traceability to their subjects in full detail.
4	Mäder and Egyed	2012	Their finding was that the performance of task on development projects with traceability was 21% quicker with task and generated 60% more accurate results or solutions.	The costs of using with and without traceability is not mentioned.
5	Bouillon, Mäder and Philippow	2013	Usage of Scenarios for RT in practice and provides insights on practitioner's traceability practices.	The researcher did not refer to some of the tools and methods that can support traceability.



Indeed, this study addressed the subject of traceability of several aspects. Firstly, the study reviewed a range of traceability practices in more details, rather than focusing on a specific technique or tool. Secondly, identified the factors that enable software companies to select a practice that fit with their work, and this is not found in previous studies. Thirdly, provided empirical evidence on the use of a set of tools and techniques in traceability software companies, where there is a lack in this area. Finally, to find out the challenges facing traceability and confirm the results of empirical evidence, the researcher conducted a series of interviews with industrial experts in this domain.

## **2.4 Summary of Chapter**

In this chapter, the explanations and details on requirements traceability are elaborated in particular, and requirements engineering in general. It also discussed the activities of requirements engineering, some of the techniques used for these activities and the importance of each activity in software development processes. In addition, the importance of requirements traceability to improve software quality, tools and techniques that used in RT, details on the challenges that facing of traceability, and prior studies related to RT practices.

## **CHAPTER THREE**

### **RESEARCH METHODOLOGY**

#### **3.1 Introduction**

Chapter three presents the steps and procedures used in achieving the objectives of this study. It defines the research methods that were used and how it helped in answering the posed research questions as stated in Chapter one.

#### **3.2 Research Procedure**

Initially, to understand the fact that studies on the specific topic does not exist in literatures, or to establish the fact that there is scarcity of research on this topic, two methods were used: appraising available primary studies and conducting a systematic literature review SLR (Khalid Khan, Kunz, Kleijnen, & Antes, 2011). Therefore, to attain the aims of this study, three approaches were adopted; systematic literature review, quantitative approach and qualitative approach.

The first is the use of a systematic review approach in the study to highlight the practices used within the software companies and extraction of the factors that enable companies to select traceability practices that are compatible with their work (in achieving the first objective). In addition, as mentioned in Section 1.3, there is scarcity of the literature that classified or gathered the traceability practices that may be used within software companies. This phase is deemed essential in this study due to fact that most of the companies in Malaysia are neglecting the traceability (Solemon et al., 2010). Thus, the systematic review concentrated on most of the practices or tools (also called “techniques”) harness for traceability within various companies in the previous empirical studies.

The second is, to determine the current practices that may be harnessed within the software companies in Northern Malaysia (second objective, see Section 1.5), the quantitative approach and more specifically, the online questionnaires were distributed to the employees of these companies. The selection of these instruments was supported by the study by Noor (2011), where an assertion was made that the most common method needed for quantitative research is a questionnaire survey. Afterwards, collected data were analyzed using Statistical Package for Social Science (SPSS) version 20.0. Indeed, these approaches are considered the practices suited to meet the objectives of this study.

The third is, to clarify and discovered the reasons which prevent software companies from implement traceability in projects development processes. The qualitative approach and more specifically, data were collected through interviews with developers who work at the software companies. Interview is one of the most widely used and powerful tools of qualitative researchers (Willig & Stainton-Rogers, 2007) In order to get a deeper understanding. Figure 3.1 summarizes the significant procedures followed to attain the aims of this study.

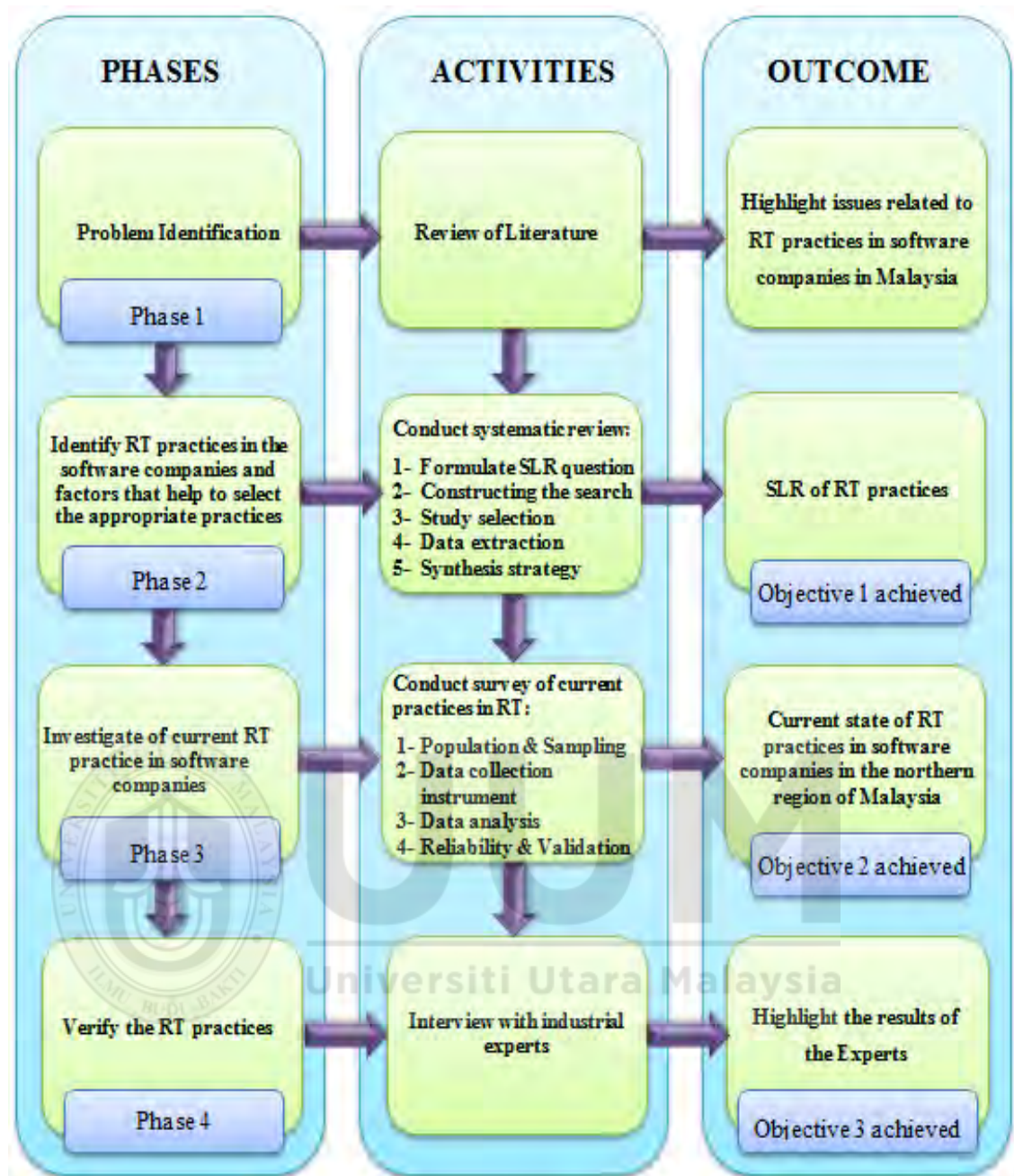


Figure 3.1. Proposed Research Framework

In general, this chapter starts by discussing the systematic literature review in more details and afterwards, quantitative approach, population and study sample as data collection and discussed. Lastly, qualitative approach, and method to collect and analysis of data are further explained.

### 3.3 Systematic Literature Review (SLR)

Systematic reviews are deemed as a method of making sense of large bodies of information, and a means of contributing to the answers to questions about what works and what does not, and also to many types of questions (Petticrew & Roberts, 2008). According to many of the researchers who asserted that, the systematic literature review is a secondary study method that has gotten much attention lately in many disciplines, such as in the studies by (Dybä, Kitchenham, & Jorgensen, 2005; Hannay, Sjøberg, & Dybå, 2007; B. A. Kitchenham, Dyba, & Jorgensen, 2004). In addition, O'Connor, Anderson, Goodell and Sargeant (2014) stated that, systematic reviews should always have a protocol. Thus, to conduct the systematic review, the researcher in this study adopted a specific protocol that was carried out based on the steps defined by Kitchenham and Charters (2007). Figure 3.2 depict the major steps of systematic review that was adopted to achieve the first objective of this study *“To identify RT practices within software companies and factors that help these companies to select appropriate practices using systematic literature review”*.



Figure 3.2. Systematic Review Guideline Process (Kitchenham & Charters, 2007)

### 3.3.1 Formulating the Research Question

Formulating the research questions which will be addressed by a systematic review is the first step in the review process, as reported by Kitchenham and Charters (2007). Hence, in this study, to enquire the traceability practices that have been used within software companies in general and factors of select these practices, the researcher formulated the following research question as mentioned in Section 1.4.

*What are the requirement traceability practices that have been used within software companies and factors that help these companies to select appropriate practices?*

In fact, systematic reviews aim to answer specific questions, rather than present general summaries of the literature on a topic of interest (Gough, Oliver, & Thomas, 2012). Thus, the research question above was formulated based on the problem statement and the PICO attributes (population, intervention, comparison, outcomes). However, since the focus of this systematic review is not to compare interventions (practices), the comparison attribute was not utilized and hence only the population, intervention and outcome (PIO) attributes of the research question were highlighted in this study. The study did not consider the comparison applied by many researchers such as Azhar, Mendes and Riddle (2012). The population of this study involves the software companies in general (public and private) which agrees with the study by Kitchenham and Charters (2007). Their study pointed out that in software engineering, the populations might be any of the following: telecommunications companies or small IT companies.

With regards to intervention, it is considered as the practices of the traceability that is harnessed within software companies. The characteristics used in describing

Intervention, were adopted from Kitchenham and Charters (2007) who defined intervention as “*the software methodology/tool/technique/procedure that addresses a specific issue*” (p.11). Finally, the outcome was discussed to shed light on the RT practices.

### 3.3.2 Constructing the Search

For the purpose of performing the search and selection of research studies related to this area, this study defined the tactic according to the date, digital libraries, and keyword to answer the research question aforementioned in the previous step. The following explains the components of this phase from the systematic review procedures:

- i. *Search Process.* In this Systematic review, the researcher concentrated on specific digital libraries available to students in University Utara Malaysia, which are:
  - a) ACM Digital Library ([portal.acm.org/dl.cfm](http://portal.acm.org/dl.cfm));
  - b) Elsevier Science Direct ([www.sciencedirect.com](http://www.sciencedirect.com));
  - c) IEEE Electronic Library ([ieeexplore.ieee.org](http://ieeexplore.ieee.org));
  - d) SpringerLink ([www.springerlink.com](http://www.springerlink.com));
  - e) In addition, the researcher also searched in the most popular search engine used for academic purposes known as Google scholar search engine (<http://scholar.google.com.my>).
- ii. *Terminology:* For the purpose of performing the automatic search of the selected digital libraries above, the researcher concentrated on the keywords (search terms) connected to this research to answer the research questions and the problem statement. The keywords in this study defined the following basic search strings:

- a) Requirement management practices OR requirement management tools  
OR requirement management technique;
- b) Requirement development practices OR requirement development tools  
OR requirement development technique;
- c) Requirement engineering practices OR requirement engineering tools  
OR requirement engineering technique;
- d) Requirement traceability practices OR requirement traceability tools  
OR requirement traceability technique;
- e) Software companies OR Software Company OR IT companies OR IT  
Company.
- f) Best practice of requirement traceability.
- g) Factors of select the RT practices OR Factors of select traceability  
techniques Or Factors of select traceability tools

### **3.3.3 Study Selection**

Various criteria were utilized in this systematic review to establish which literature will be inclusion or exclusion. The points listed below illustrate exclusion criteria:

- i. Studies that are irrelevant to the research questions;
- ii. Studies outside the indicated time period, the time span from 2008 to 2015 for systematic review of this study is considered a complement study for the prior literature. Where the latest study was in 2007, there are important new tools for traceability. These new techniques would enhance the performance and products for companies or organizations. Hence, highlighting new tools and techniques are crucial for this study. This systematic review specifically focused on the practices (referred to as techniques or tools in some studies, practice will be used as a term throughout the research). In fact, the main question for the systematic



review for this study is to identify the practices of RT, while previous studies mainly concentrated on the challenges;

- iii. Studies that do not determine or specify the practices, tools, or techniques;
- iv. Studies lacking empirical basis for their findings;
- v. Duplicate studies from varying sources; and
- vi. Studies not reported in English.

#### **3.3.4 Data Extraction**

The objective of this step is to create data extraction forms to accurately document the information obtained by the researcher from the primary studies. Data extraction also involves presenting a detailed table which describes all the study that is reviewed in detail (not every study that was located in the review, only studies meeting all the inclusion criteria that will be highlight in previous step) (Petticrew & Roberts, 2008). In most cases, data extraction define a set of numerical values that should be extracted for each study. However, numerical data are important for any attempt to summarise the results of a set of primary. This study strived to adopt data extraction strategy from Salvador, Nakasone and Pow-Sang (2014). In brief, the content of the data extraction form included the following information: (a) Study title, (b) Author (s), (c) type of publication, (d) extraction date, and (2) database in which the study was found.

#### **3.3.5 Synthesis of the Extracted Data**

The last step from the systematic reviews strategies is called "*Synthesis*". This step involves collating and summarizing the results of the included primary studies. Generally, the systematic review was harnessed to support the problem statement, due to the lack of discussions on traceability in specific, and also to understand the current practices that has been used within the companies whether in public or private sectors.

### 3.4 Quantitative Approach

The literature has identified the quantitative approach as the most suitable approach for investigating the individual opinions and the motives behind the actions, behavior, and attitudes of respondents. According to the study by Smith (2012), the quantitative research approach can validate the conclusion of the study by verifying the established concept and by proving or disproving a proposed concept. In the same context, Kumar (2011) and Atieno (2009) also identified the quantitative research approach as the best scientific research method given its precise measurements via deductive approach and its employment of measurable data collection tools. This study employed this approach as a method of gathering the data to attain the second research question:

*What are the current practices of requirement traceability that applied within software companies of Malaysia?*

#### 3.4.1 Populations and Sampling

Muijs (2010) defines the population as the group of people we want to generalize to. In the same vein, Balvanes and Caputi (2001) stated that populations must be accessible and quantifiable and related to the purpose of the research. This study, as referred in the problem statement section, concentrates on the requirement traceability practices, especially the software companies in the northern region of Malaysia. Thus, population is defined according to the objectives of the study.

With regards to sample, there is in general no correct sample in the absolute sense of the matter. Some researchers suggest that larger samples are always preferable (Charreire & Durieux, 2001). Others, however, argue that a large sample is no guarantee of accuracy (Balvanes & Caputi, 2001). This is also asserted by the study by Noor (2011), where it is stated that:

*“There are no strict rules to follow [mean for sampling], and the researcher must rely on logic and judgment” (p.177)*

In general, this study adopted the snowball sampling method, since the number of software companies in the northern region of Malaysia is not fixed. Snowball sampling is a special non-probability method used when the desired sample characteristic is rare (Aartsengel & Kurtoglu, 2013). In snowball sampling, the researcher identifies one or more key individuals who name others that might be candidates for the specific research (Creswell, 2012). In this study, snowball sampling is more appropriate because it reduces cost dramatically. Also, it is often embraced as the only way to reach hidden populations, and attain more accurate and credible results as a result of the participants being identified by people who have the experience and knowledge. Actually, snowballing is considered a useful technique in sampling which has been adopted by many researchers in industry/company's studies (Bonaccorsi, Giannangeli, & Rossi, 2006).

Likewise, Creswell (2012) stated that when selecting participants for a study, it is important to determine the size that will be needed by the sample researchers. Therefore, the sample size for this study is determined through the rule of thumb, which states that the sample must include between 30 to 500 respondents (Sekaran & Bougie, 2010). In this study, data was collected from (31) practitioners in software companies from different parts of the northern region of Malaysia (Please refer to Appendix C).

### **3.4.2 Data collection Instrument**

Probably the most popular (quantitative) research design in the social sciences is survey research (Muijs, 2010). Also, Smith (2012) reported that questionnaires are better than most data collection instruments because of their inexpensiveness and anonymity. In

addition, according to Harfoushi et al. (2012), questionnaire is an efficient way to collect data in scientific researches because it is an easy, effective and inexpensive method. Therefore, this study used the online questionnaire based on previous studies and according to the results from the systematic review, to elicit the current practices that are used within the software companies in Malaysia.

In this study, the questionnaire consists of three parts. The first part is about participant's background such as company name, position, email, age, gender, educational level, organization level, sector of the software company, the number of years in experience, number of employees in the company, and the number of projects involved in. The second and third parts of the questionnaire consists of a set of questions on the use of tools and techniques to trace requirements respectively. Meanwhile, the second and third sections contains 14 questions adapted from Rochimah et al., (2007), Shahid et al., (2011), Winkler and Pilgrim (2010), Cleland-Huang (2005), Mäder et al., (2008), and Cleland-huang et al., (2009) to collect data on the use of traceability practices for tracing requirements in software projects. For every question, a scale between 1 and 5 is created as responses with 1 = Never , 2 = Rarely , 3 = Sometimes , 4 = Often, and 5 = Always, which is based on the study by (Vagias, 2006). Table 3.1 illustrates the questionnaire sections and its sources (Please refer to Appendix A).

Table 3.1

*The sections of questionnaire and the sources of the adapted questionnaires.*

Section	No. of Questions	Code	References
<b>Demographic Information</b>	12	Q1	Literature Review
		Q2	
		Q3	
		Q4	
		Q5	

		Q6	
		Q7	
		Q8	
		Q9	
		Q10	
		Q11	
		Q12	
		Q13	
<b>Requirements</b>	5	Q14	(Cleland-Huang, 2005;
<b>Traceability Tools</b>		Q15	Mäder et al., 2008;
		Q16	Muhammad Shahid et al.,
		Q17	2011)
		Q18	
		Q19	
		Q20	(Cleland-huang et al., 2009;
<b>Requirements</b>	9	Q21	Rochimah et al., 2007;
<b>Traceability</b>		Q22	Winkler & von Pilgrim,
<b>Techniques</b>		Q23	2010)
		Q24	
		Q25	
		Q26	

### 3.4.3 Data Analysis

After data was collected from a representative sample of the population, the next step was to analyze this raw data to attain the objective of this study. Indeed, this study does not focus on variables or hypothesis, or also the relationship. It used the advanced statistic software to analyze the raw data. Generally, the study strived to utilize descriptive statistics. Descriptive statistics is particularly useful in communicating the results of experiments and research. Moreover, Abdalla, Bourse, De Caro and Pointcheval (2015) stated in their study that, descriptive statistics is the discipline of quantitatively describing the main features of a collection of information.

#### **3.4.4 Validation**

With regards to validation, this study harnessed the content validation where the instruments were distributed to three experts (Please refer to Appendix B). Face validity assured that these instruments provides adequate coverage for a topic. This was suggested by (Noor, 2011; Sekaran & Bougie, 2010), who pointed out that expert opinions help to establish content validity. Their feedbacks were documented. They suggested just a few minor changes or corrections on terminologies, coordination of questions, and editorial works. Therefore, all recommendations were performed.

#### **3.4.5 Reliability Analysis**

Before gathering the actual data from the respondents, a pilot test was conducted for the data collection tools and procedures. The benefits of conducting a pilot test includes the identification of errors, testing how long it takes to complete, to check that the questions are not ambiguous, to check that the instruments are clear, and to make corrections to the questionnaire. Malhotra (2008) indicated that the least number of respondents that are appropriate for a pilot test to validate the questionnaire ranges from 15 to 30 respondents. In the same vein, Hill (1998) suggested 10 to 30 participants for pilots in a survey research. Hence, this study used 15 participants from some software companies and UUMIT for the pilot test. This pilot test was distributed in mid-January 2016.

The reliability of this study was conducted after the pilot study, to ensure the questionnaires will function effectively (Noor, 2012; Kothari, 2009). According to Sekaran and Bougie (2010), the reliability test is established by testing for both consistency and stability. The consistency test shows how well the items measure concepts together as a set. Cronbach's alpha that generates from this test is a reliability coefficient which indicates how well the items in a set are positively correlated to one

another. The Cronbach's alpha is computed in terms of the average inter correlations among the items measuring the concept. Based on Table 3.2, the result shows that the alpha value in this study is 0.980, which is considered as very good.

Table 3.2

*Pilot test Cronbach's Alpha*

Cronbach's Alpha	N of Items
.980	14

Furthermore, Sekaran and Bougie (2010) stated that reliabilities less than 0.60 are considered to be poor, those in the 0.7 range are acceptable, and those over 0.8 are considered good. In addition, Coakes and Steed (2009) reported that alpha more than 7, is very reliable.

### 3.5 Qualitative Approach

Qualitative research explores a wider scope where modification occurs and is capable of capturing the complete set of factors perceived by participants as a contribution to outcome or change (Bauer & Gaskell, 2000). It is mostly utilized for depth than breadth of information (Chisaka & Vakalisa, 2000). It is a method for learning and knowing about varying experiences or practices from the view of the people involved. Additionally, Creswell (2012) identified that in qualitative research, the objective is to look deeper into the complex set of elements around the core phenomenon and bring forth the different meanings or views that participants have. Hence, qualitative research permits researchers to look into social phenomenon and what it means in everyday life (Burns & Grove, 2003; Speziale & Carpenter, 2003) as compared or related to the quantitative approach described in Table 3.3.

Table 3.3

*Differences between quantitative and qualitative approach (Bauer & Gaskell, 2000)*

	Qualitative	Quantitative
<b>Data</b>	Texts	Numbers
<b>Analysis</b>	Interpretation	Statistics
<b>Prototype</b>	Depth interviewing	Opinion polling
<b>Quality</b>	Soft	Hard

In addition, Richards and Morse (2012) recommend the use of the qualitative approach in their statement “*if the purpose is to understand an area where little is known or where previously offered understanding appears inadequate, you need qualitative method that will help you see the subject anew and will offer surprises*”. The purpose of qualitative approach is to verify and reconfirm the outcomes of survey by conducting a series of interviews with experts in this field. In other words, to understand the reasons behind less use of the RT practices among the software companies, this study utilized this approach as a method of gathering the data to achieve the third research question:

*How to verify the requirement traceability practices used by the software companies?*

### 3.5.1 Interview

In this study, the gathering of data was done by interviewing developers who work at the software companies (these individuals have good experience in this domain) as suggested by some related studies (Fontana & Frey, 2005; Marshall & Rossman, 1999). Technically, Ackroyd (1992) reported that “interview encounters between a researcher and a respondent in which an individual is asked a series of questions relevant to the



subject of the research". That is to say, a qualitative interview happens when researchers ask a number of participants open-ended, general questions and take their feedbacks into record (Creswell, 2012). This is one of the most widely used and powerful tools of qualitative researchers (Willig & Stainton-Rogers, 2007). Furthermore, during interviews, better control is assumed by the interviewer over the nature of information received as a result of the interviewer being able to make specific queries to bring about specific information (Creswell, 2012). Also, majority of the questions for interviews in this study have been adapted based on previous studies, and according to the results from the systematic literature review (e.g. Egyed, Grünbacher, Heindl, & Biffl, 2009) and (Please refer to Appendix D).

Additionally, the interview questions were designed to lead the interviewees towards a goal of helping the researcher to identify the answers to the research question as well as to confirm the results of questionnaire (The second objective). The first question is a general question to explore whether traceability practices applied in the software development projects at software companies or not. Meanwhile the remaining contains questions that highlight the issues or challenges faced by practitioners in applying traceability practices, in addition to details about the importance of traceability practices, as well as the participants' recommendations. After discuss the interview questions with the supervisor, the questions were validated by experts in qualitative research (Dr. Nassir Jabir Farhan) to strengthen the structure of the questions and understand how to extract rich data from participants.

### **1.5.2 Data Analysis**

The purpose of the interviews is to find out the reasons for the weakness of the use of traceability practices in software development processes by the viewpoint of those experiencing it. For the analysis of data, this study used the coding method to analysis

of interviews data. Coding is one of the numerous techniques in increasing knowledge about data in a form of words or few words phrases which symbolically allocates a salient, summative, evocative, and/or essence-capturing feature for a percentage of visual data or language-based (Saldaña, 2015). In general, this study was coded the short phrase of the interview transcripts based on the ended-open questions. In addition, Miles and Huberman (1994) referred that, the majority of qualitative researchers will code their data both during and after collection as an analytic tactic, for coding is analysis. This argument also mentioned by Basit (2003), who stated that, coding and analysis are not synonymous, though coding is a crucial aspect of analysis. However, in the present study the researcher used the questions as a code. This manner for coding the raw of the data recommended by Auerbach and Silverstein (2003), they said that "you <researcher> keep a copy of your research concern, theoretical framework, central research question, goals of the study, and other major issues on one page in front of you to keep you focused and allay your anxieties because the page focuses your coding decisions" (p.44), as well as this recommended agreed by (Saldaña, 2015).

### **3.6 Summary of Chapter**

The methodology of the research is presented in this chapter. Several procedures and justifications are incorporated in the methodology to fulfill the objectives and to answer the questions of the research. The research framework is also presented in this chapter. To attain the objectives of this study, three approaches were adopted for this purpose; systematic literature reviews, quantitative and qualitative approach which are sufficient for gathering rich data to achieve the objectives discussed previously.

## **CHAPTER FOUR**

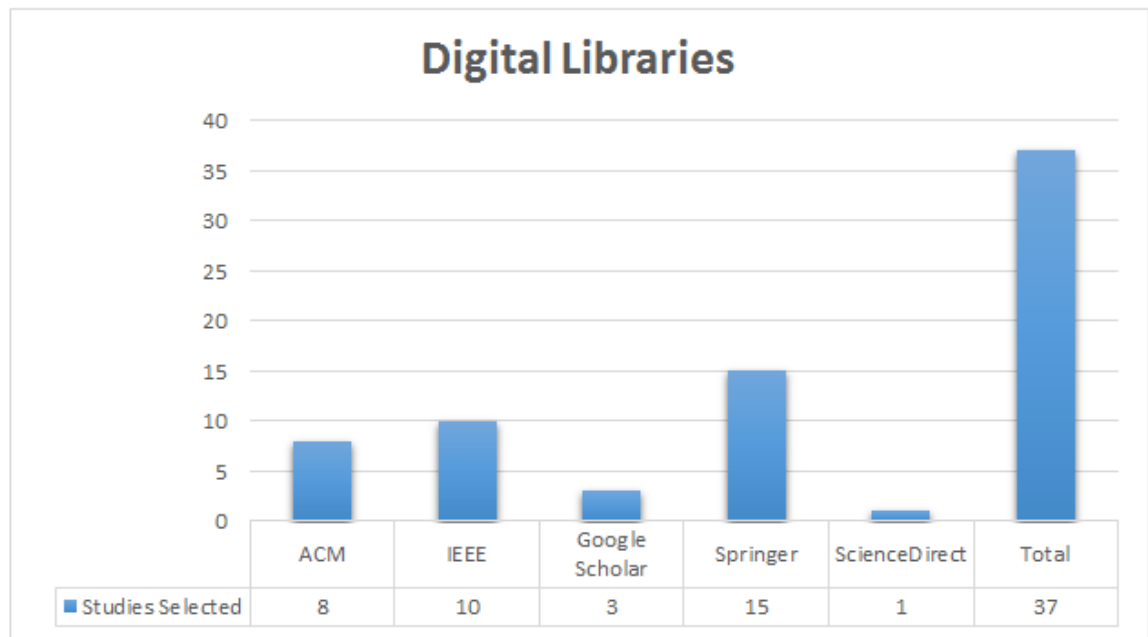
### **RESULTS OF SYSTEMATIC LITERATURE REVIEW**

#### **4.1 Introduction**

This chapter presents results of systematic literature review that includes traceability practices used by software companies. It includes the techniques and tools used to support requirements traceability during software development. All results were based on the methodology or the steps that have been discussed in the previous chapter.

#### **4.2 Conducting the Review**

This section focuses on conducting the review by implementing the SLR protocol planned and discussed in chapter three. To generate the first pool of articles, citation search was done with the exclusion of additional constraints. Identification of the overall 209 citations have been realized in this initial step via the search strategy. Subsequently, each of the citations were reviewed by the researcher and a set of citations that could be significant by reading the abstracts, introductions, and conclusions, were selected. The results from this round was 95 studies. More specifically, after reading all 95 studies to identify practices of RT, the inclusion criteria (for this stage) was met by 37 of 92 studies. The details concerning the amount of selected studies discovered while conducting the search, are presented on Figure 4.1, and the complete number of selected studies is shown in Appendix D.

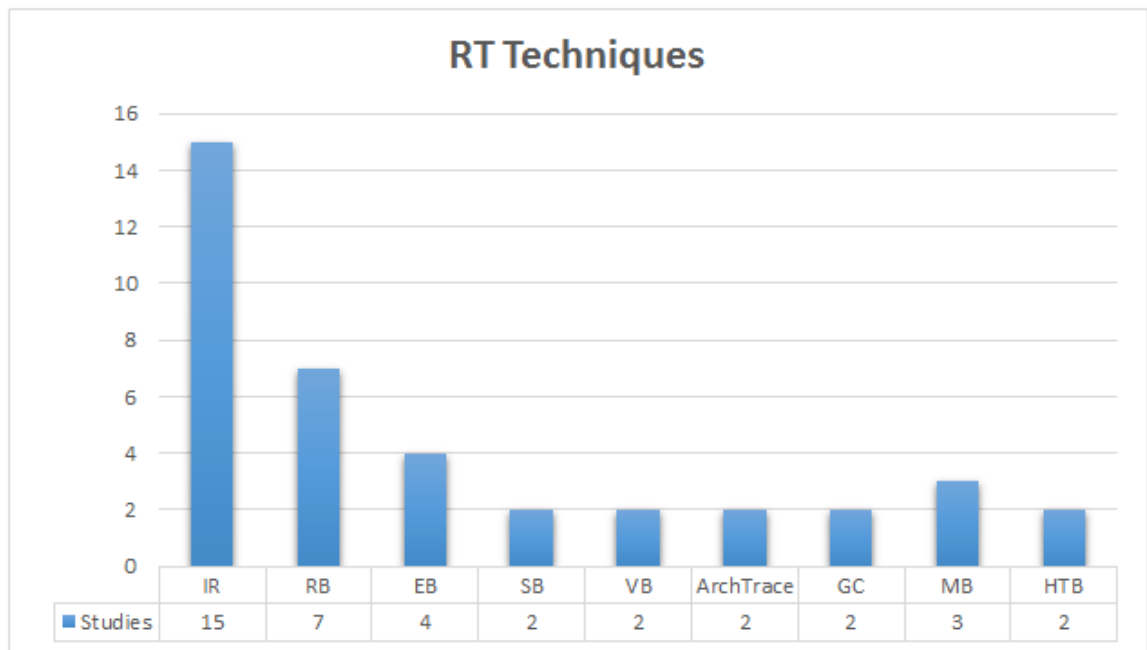


*Figure 4.1. Studies Selected of Digital Libraries*

Figure 4.1 shows the studies that have been selected from the digital libraries. The total number of selected studies is 37 from various digital libraries. 15 studies were taken from Springer, while 10 and 8 studies were taken from IEEE and ACM respectively. 3 studies were obtained from Google scholar, and only one study from ScienceDirect.

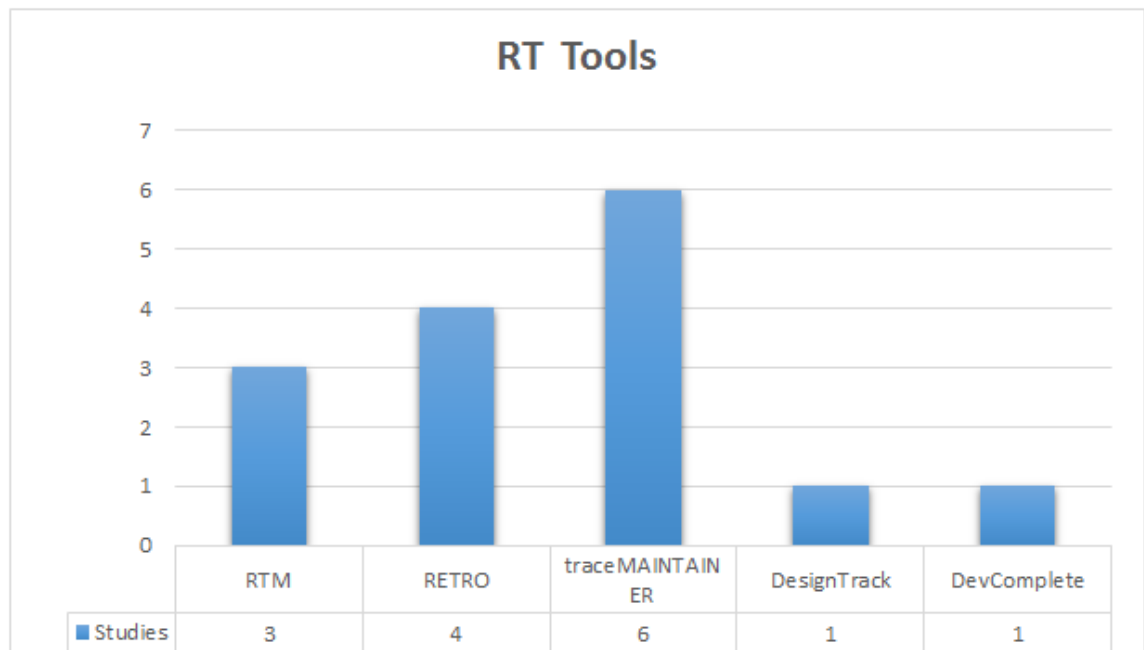
### **4.3 Result Of SLR**

To understand what are the practices used to trace requirements during software development life cycle, SLR questions were established to identify the techniques and tools used in RT. The research questions were answered by examining the literature on RT. Based on the results from the SLR process, the following techniques were used with requirement traceability according to included criteria: Rule-Based (RB), Value-Based (VB), Information Retrieval (IR), Scenario-Based (SB), Event-Based (EB), Hypertext-Based (HTB), ArchTrace, Model-Based (MB) and Goal-Centric (GC), as depicted in Figure 4.2.



*Figure 4.2. RT Techniques for Tracing Requirements*

With regards to RT techniques, the majority of the studies were about IR technique which included 15 studies. 7 studies contained information about the rules-based technique. As for the other techniques, information pertaining to them were taken from studies within the range of 2 to 4 studies. In addition, the researcher found the following traceability tools during the review of the literature: Requirement Traceability Matrix (RTM), RETRO, traceMAINTAINER, DesignTrack and DevComplete, as depicted in Figure 4.3.



*Figure 4.3. RT Tools for Tracing Requirements*

Figure 4.3 above shows the number of studies that contained information about requirements traceability tools. 6 studies included information on traceMAINTAINER tool. While 4 studies included information on RETRO, 3 studies contained details on RTM tool. Finally, only one study covered DesignTrack, and a similar result was obtained for DevComplete as well.

#### **4.3.1 Requirement Traceability Techniques**

This section explains each of the RT technique identified during the SLR.

##### **a) Rule-Based (RB)**

Spanoudakis et al., (2004) presented a RB technique used in extracting traceability links between requirement statement document, analysis object models, and use case documents. They used mainly two rules, i.e. inter-requirement (IREQ) and requirement-to-object model (RTOM).

RTOM rules are used in tracing syntactically related terms (or words) in the textual part of requirement statement document and use case documents to an analysis object model. When RTOM rule matches are found, a rule specified type traceability link is created. The syntactic traces are described as patterns of terms having specific grammatical roles in a piece of text. These grammatical roles are set by probabilistic grammatical tagging methods. IREQ rules are utilized when tracing between use case documents and requirement statement documents. These traces are called derived traceability since they are created between parts of the use case documents and the requirement statement document. They will only be generated, if the parts have been traced to elements of analysis object models by RTOM traceability link (Ali, Guéhéneuc, & Antoniol, 2012; Ali, 2010).

RB technique requires the export of all supported artifacts into the eXtensible Markup Language (XML) format, and the rules generate traceability relations for the exported state of the models (Zisman, 2012). RB traceability comprises of four stages (Ali, 2010):

- i. The grammatical tagging of use case document and requirement statement document using CLAW; CLAW is a form of grammatical tagging used in tagging the 100-million-word British National Corpus (BNC) of which 70 million words were tagged at the time of writing.
- ii. The conversion of use case document, requirement statement document, and analysis object models into XML representation.
- iii. The generation of traceability links between use case document, analysis object models, and requirement statement document. And,
- iv. The production of traceability relations or associations between varying parts of the use case documents and the requirement statement.

RB technique does not support partial generation for artifacts evolution. If change has been applied to a certain artifact, then traces will be re-generate by RB technique as it produces links for the first time (Mäder et al., 2008).

#### **b) Value-Based (VB)**

Heindl and Biffel (2005) proposed a VB technique for RT. This technique provides an economic model and a technical model for requirements tracing. It is a framework for accessing or measuring the worth that requirement traceability can deliver to a company or an organization for the purpose of supporting decisions associated to the implementation of requirement traceability. VB approach considers an economical and a technical model built on a number of criteria such as number of requirements, risk of requirements, value of requirements, number of artefacts, precision of traces, number of traces, effort of trace maintenance and identification, cost, size of artefacts, and value of traces (Heindl & Biffel, 2005).

The value based approach integrates a semi-automated and a manual method in attaining the traceability link, and also in executing an alteration in the software artefacts (Ali, 2010). Again, the VB technique offers traceability analysis in addition to change impact analysis for supporting software evaluation. Furthermore, this technique can determine requirements that are very important to trace, and also requirements that are less valuable to trace (Egyed et al., 2009). VB consist of five processes (Ali, 2010):

- i. Requirement definition: requirement engineer or project manager analyze each requirement and assign a unique identifier to create a requirement list with id.
- ii. Requirement prioritization: all stakeholders asses the requirement and divide them on three priority levels based on effort, risk and value of each requirement.



- iii. Requirement packaging: identification of requirement cluster to define and refine the architecture from a specific set of requirements.
- iv. Requirement linking: project team set traceability link between requirements and varying software artefacts. (Significant requirements are traced in detail whereas less significant requirements are traced with less details in order to create the overall traceability plan)
- v. Evaluation: project manager use traces in estimating the impact or effect of requirements change.

### **c) Information Retrieval (IR)**

Many researchers have tried to use IR technique to establish traceability link between different software artefacts (Gotel & Finkelstein, 1994; Lin et al., 2006). IR technique concentrates on automating the creation of traceability link using comparison based on similarity between two (2) types of artifacts (Ali, Sharafi, Guéhéneuc, & Antoniol, 2015). This technique is normally based on natural language text documents and independent of structural properties. It uses an indexing process on the collection of pre-processed documents (corpus) (Ali, 2010). This is done in the following steps (De Lucia, Marcus, Oliveto, & Poshyvanyk, 2012):

- i. Extracting all the terms from corpus;
- ii. Removing stop words and stemming;
- iii. Developing a dictionary of all the terms and number of times it occurred in that document;
- iv. Calculating term frequency at document level;
- v. Calculating inverse document frequency to reduce the weight of most frequent words, at the corpus level;

- vi. Developing (calculating terms frequency and calculating inverse document frequency) weight matrix; and
- vii. Computing similarity between document and query to retrieve a ranked list of matched documents.

The user can utilize the ranking to understand the relationship between artifacts and even requirements in order to validate the links generated by the IR technique. This is achieved by reviewing each link in order to accept or reject; an activity which remains a manual task that cannot be automated (Ali, 2010). The most commonly used IR models applied to traceability are: Latent Semantic Indexing (LSI), Probabilistic Model (PM), and Vector Space Model (VSM) (Chen, Hosking, & Grundy, 2011).

The efficiency of IR technique is computed using two main metrics: precision and recall. For a given query, recall is the percentage of actual retrieved links and precision is the percentage of correct links as a ratio to the total number of traces retrieved. One of the disadvantages of using IR is the human intervention for low precision queries (Bacchelli, Lanza, & Robbes, 2010). IR have been implemented on different tools, with RETRO as an example of such tools (Cuddeback, Dekhtyar, & Hayes, 2010; Pandanaboyana et al., 2013). The scope of tracing covers almost all artifacts including high-level and low-level requirements, design elements, manual documents, source code, and test cases (Zou, Settimi, & Cleland-Huang, 2010).

#### **d) Scenario-Based (SB)**

Egyed and Grunbacher (2002) proposed a SB approach. SB uses an assumed trace information that has to be entered manually. After this, it utilizes runtime information to generate trace links. SB has three pre-requisite in order to create traceability link between artefacts. Firstly, there must be an executable or simulate-able software system

which can be either partial implementation or incremental prototype. Secondly, a software model must be available for the system. Finally, there must be executed test cases or scenarios (Ali, 2010).

Even though the traceability links are generated automatically, the hypothesized trace information must be entered manually (semi-automatic). Using this technique (SB), the creation of traceability links can only be done with an available running system (Ali, 2010).

SB technique can be used in validating the four types of traces; traces between model elements and system, traces between scenarios and system, traces between models element, and traces between scenarios and model elements (Ali et al., 2012). SB can help to minimize ambiguity in manual links by establishing dynamic verification in large systems. Furthermore, SB is a feasible solution for requirements modification management activities in addition to impact analysis of changing requirements (Ali, 2010).

#### **e) Event-Based (EB)**

Cleland-Huang et al., (2002) proposed the EB approach for managing and maintaining traceability between requirements and UML along with test cases. EB is a semi-automated procedure for gathering requirement traceability (RT) links for relating developers and code artefacts to use cases. The production of trace links were done by monitoring events started by a developer operating in the context of a use case, on a code artefact (Omoronyia, Sindre, Biffi, & Stålhane, 2011). Compulsory inputs to this technique include: requirements, UML artifacts, and test cases (Ali et al., 2012).

EB technique supports two type of changes namely, functional and contextual change. If a quantitative change occurs in existing requirements, it is referred to as contextual

change. The contextual changes update the EB system and conduct automated re-execution of significant performance models with the new values from the modification. It enables project managers to decide on whether new changes should be implemented or not, and also helps developers to understand change effects. Functional changes occur when a new requirement is added in the system. it involves the functionality and performance which makes them hard to predict (Ali, 2010).

EB technique involves three main components: (a) Requirement manager, (b) Event server, and (c) Subscriber manager. When a change occurs in requirements (like refinement, replacement, abandonment, or merge), an event notification message is published. The event server receives event notification and forwards to every dependent artefacts. It is the responsibility of the subscriber manager to listen to all incoming event notification for a set of similarity typed artefacts on behalf of the subscribers which it oversees or manages the event notification sent by the event server. In certain cases, the managed object may be automatically updated by the event manager, while in other cases, the event message must be deposited or stored in an event log until manual intervention occurs (Ali, 2010; Mäder & Gotel, 2012). The main merit of EB is that when modifications are introduced into the system, they can be efficiently disseminated throughout the system of artefacts (Ali, 2010).

#### **f) Hypertext-Based (HTB)**

Maletic et al., (2003) proposed the hypertext-based for traceability between different artefacts. HTB is a technique used for traceability link generation using open hypermedia and information integration (Sherba, Anderson, & Faisal, 2003). This technique is semi-automated, hence, user input is necessary for different tasks (documents, selecting source code, selecting dimensionality subspace of LSI, and determine threshold value for traceability) during the HTB process (Ali et al., 2012).

Each file is treated as a document in this technique, and if the size of the files are too large, it fragments into parts which are roughly the size of an average document in the corpus (Ali, 2010).

The HTB technique uses XML as the key tool for the representation of created links and models. The documents are generally categorized into two: conformance and non-conformance. Conformance category is further divided into two types; the first type is causal conformance and the second is non-causal conformance. The causal relationship carry signified logical ordering or arrangement of documents involved, while a non-causal conformance relationship occurs when the documents or some parts of the documents must conform with each other. In non-causal relations, the causality cannot be distinctly recognized. The non-conformance links are meant to anchor organizational and navigational associations that have little or no significance to the determination of agreements between documents (Ali, 2010).

#### **g) ArchTrace**

Murta, Van Der Hoek, and Werner (2006) presented the ArchTrace technique for the advancement of traceability connections between the architecture and implementation. This technique delivers tool support for versioning the relationships between a component-and-connector architecture and its execution as the architecture or code evolves with time (Murta, van der Hoek, & Werner, 2008).

In ArchTrace, this technique provides a policy-based infrastructural to perform automatic update between traceability link architectural and implementation. ArchTrace differentiates itself by frequently updating traceability relations or links from architectural elements to code elements via a policy-based extensible infrastructure which permits developers or a group of developers to choose a set of traceability

management policies that is suitable for their situational needs and working styles (Hong, Kim, & Lee, 2010).

ArchTrace can be characterized as an immediate update which depends on two critical observations: The first is, rather than regenerate traceability links following the passage of a significant amount of time, a constant update of the links is performed in reaction to every modification made by a user. The second observation is that the precise update that will be performed is decided by a keenly identified group of policies for traceability management. The outcome is a technique which can be customized to varying user practices, taking advantage of the information encrypted in the policies concerning source code and architectural evolution, and also considering the addition of new policies (Murta et al., 2008).

#### **h) Model-Based (MB)**

Cleland-Huang, Hayes, and Domel (2009) proposed a MB technique for RT. Various organizations spend significant effort and cost in constructing traceability matrices that fulfill the process improvement initiatives or regulatory requirements. Unfortunately, the constructed matrices are often left out of use and project stakeholders keep executing serious software engineering activities like requirements satisfaction assessment or modification impact analysis without the advantages of the established traces. This is as a result of the absence of a process framework and related tools to back the utilization of these trace matrices in a tactical manner.

The purpose of the MB technique is to determine artifact granularity suitable for traceability and to extract all candidate traceability links in order to achieve a recall rate of 100% (Ohashi, Kurihara, Tananaka, & Yamamoto, 2011). Model-based techniques are developed to assist organizations obtain complete advantage from the traces

constructed and to permit the project stakeholders to plan and produce trace schemes, and to execute them in a modeling environment which is in a graphical form. MB approach consists of a standard notation for presenting strategic traceability resolutions in a graphical form, and also notation for presenting reusable trace query modeling via sequence diagrams which have been augmented. XML is used in representing project specific data and every other model elements. There are four different base layers in MB technique (Cleland-Huang et al., 2009):

- i. The strategic layer: This layer presents the artifacts and related traceability links in a model called the strategic traceability graph. The graph assumes the structure of a traceability metagraph that determines what kind of artifact should be traced, and identifies link types with more information regarding users of the link, where data is stored, and when the link will be used. The aim of strategic traceability graph, is to come up with decisions as to the amount of traceability their project requires, and to survey the aim and estimated merits of their traceability plan.
- ii. The document management layer: In this layer, the locations and names of every traceability matrices and project level software artifacts, are recorded. These constituents are denoted as XML documents in the initial stages of the approach.
- iii. The stored query layer: The stored query layer determines a group of queries which are aided by the document management and the lower-level strategic layers. The queries are developed by a requirements analyst or project manager and utilized by the other stakeholders during their usual software engineering activities.
- iv. The executable layer: This layer is in charge of interpretation of the queries in the stored query layer. It is also in charge of transforming the queries into

executable codes and then invoking them when needed. Visualization or reporting of results are also performed in this layer.

The proposed technique needs additional effort beyond and above the significant effort already required to build and preserve traceability links. Though, its advantages are appreciated with traceability queries made more available to additional stakeholders, and also when the queries as well as related strategic traceability graphs are used again in successive models. MB possess the prospective to considerably increase the merits which can be obtained by creating traceability matrices for projects (Abbors, Truscan, & Lilius, 2009; Cleland-Huang et al., 2009).

#### **i) Goal-Centric (GC)**

Cleland-Huang et al., (2005) proposed the GC approach for trace non-functional requirements (NFRs). It provides traceability anchorage for maintaining and managing (NFRs), and associated quality concerns through the long life span of a software intensive system.

GC supports two precise traceability activities. The first includes recognizing the initial impact of a modification on the GC model. For instance, if a segment of the code is changed by a developer, or a change is made in a certain UML model by an architect, traceability is utilized to identify or recognize possible operationalization (design solutions) or related goals in the goal model. This initial trace can be performed using either automatically generated or manually created traces. The second traceability activity is activated immediately the first impact point is found. The Goal-Centric framework consist of (Mirakhorli & Cleland-Huang, 2012):

- i. A goal model which encompasses stakeholders quality or value concerns and their compromises;



- ii. A set or group of QAMs designed to assess the degree to which the identified quality goals have been satisfied by the architecture;
- iii. A traceability infrastructure which is used in connecting goals to QAMs;
- iv. GC algorithms which controls the propagation of modification through the goal hierarchy and the automated impact analysis; and
- v. An impact documentation or report that explains the possible effect of a modification on the general quality goals.

The advantages of this technique are that they deliver greater degrees of automation for utilizing and comprehending traceability links, and in certain cases are developed precisely with maintainability in mind (Mirakhorli & Cleland-Huang, 2012).

#### **4.3.2 Requirement Traceability Tools**

This section will cover the RT tools that were identified during the review of literature.

##### **a) Requirement Traceability Matrix (RTM)**

RTM are mostly utilized in industries to determine the relationships between requirements and other kinds of artifacts such as test cases, code modules, and design (Cuddeback et al., 2010). More specifically, it is a document that links requirements with the conforming test cases. Each of the requirements are itemized in a matrix row, and the matrix column are used to locate where and how each of the requirements have been addressed. It is also utilized in the management of modifications and as the base for test planning. Preparing traceability matrix ensures that every functionalities required of the application in the test cases has been covered. A good traceability matrix also provides ease in track changes. Hence, the quality of a system is improved (Athira & Samuel, 2011).

RTM comes with ease of comprehension and can be utilized in simple scenarios (even by non-technical users), such as checking if single links exist between artifacts, or taking record. Though for projects based on real-world cases, traceability matrices may become very enormous and hence, unreadable. The content of each artifact is normally concealed in the matrix, but referenced with unique identifiers. This distinction of the grid's two-dimensional nature, makes it challenging to track links across numerous artifacts recursively. Lastly, it is nearly not possible to represent n-ary links on a traceability matrix of two-dimensions in a comprehensible manner (Winkler & von Pilgrim, 2010). RTM is an approach used in validating a product's compliance with the product's requirements (Athira & Samuel, 2011).

#### **b) RETRO**

Tracing requirements has been a significant process of the software development life cycle. For instance, it is important to know if all the requirements for the product developed have been satisfied. Again, traceability information is needed for the performance of impact analysis on all projected modifications (Pandanaboyana et al., 2013). However, requirements tracing by manual traceability methods are prone to errors and also time-consuming (Cuddeback et al., 2010). Requirements tracing on-target (RETRO) is an RT tool presented by Hayes et al., (2007) to ease the automatic creation of requirements traceability matrices (RTM). RETRO utilizes IR and text mining methods to construct candidate traces. Result revealed that RETRO discovered considerably more accurate links than manual tracing (RTM) and used only one third of the usual time to do so (Shahid et al., 2011).

Where there occurs a great difference in the documents to be traced because they have been developed by different organizations, utilization of the thesaurus offers a more efficient tracing method. For instance, an organization may use the term 'failure' in the

stance where another organization will use the term ‘error’. These two words will not be related by IR technique to be having similar context. The thesaurus option, provided by RETRO, is used in overcoming this issue (Pandanaboyana et al., 2013).

### c) **traceMAINTAINER**

traceMAINTAINER is a prototype tool which anchors the semi-automated update of traceability connections between analysis, requirements, and design models of the software systems which are expressed in UML (Mäder, Gotel, & Philippow, 2009c). It is a tool that supports a technique for preserving post-requirements traceability connections or relations after modifications have been applied to the traced model elements (Mäder, Gotel, & Philippow, 2009b). Preservation of traceability relations is founded on predefined rules. A development activity is recognized by each rule, which is applied to a model element (Mäder et al., 2008). traceMAINTAINER allows the update of traceability relations by developers, with little manual effort (Mäder et al., 2008; Mäder et al., 2009c).

Microsoft Visual Studio.Net has implemented the traceMAINTAINER software which supports the following common scenarios: the examination of events change flows based on a set of rules which are predefined and imported from an XML rule catalog; the implementation of essential traceability updates following the recognition of development tasks; and the validation and edition of already existing or current rules alongside the establishment of new rules (Mäder et al., 2009c).

The drawback of this tool is that only activities that have been predefined can be recognized at the moment. This does not reflect all the approaches for possible developments, hence, it makes rules customization essential to project specifics. While an initial cost exist in the identification of development activities and rules formulation,

the set of rules has proven to be reusable in the regulated domain of UML-based object-oriented software engineering. Future work is to obtain additional statistical data on the benefit/cost trade-off, benefits with regards to the time minimized on manual maintenance through all projects utilizing the rules, and costs with regards to the initial definition of the rules (Mäder et al., 2008).

#### **d) DesignTrack**

DesignTrack as a traceability tool is a prototype tool which provides an incorporated design environment for form exploration and requirement specification in one design session (Ozkaya & Akin, 2007). It offers an environment for the navigation of intricate design information spaces through supporting requirement traceability. It also calculates the power, applicability, and limitations of computer-aided designs which are enabled by requirement traceability. In addition, this tool can be utilized in the management of design requirements issues. Shahid, Ibrahim and Mahrin (2011) stated that, this tool has not been evaluated experimentally yet.

#### **e) DevComplete**

DevComplete is a traceability tool designed by SmartBear software. It offers a complete traceability for project tasks, requirements tracking, and flaws for team agility improvement. Using a clear and comprehensible map which presents the requirements, defects, and project tasks, it can execute the workflow to prevent requirements from being modified after base-lined and approved. DevComplete permits freedom to implement requirement modifications according to current demands. It also detects the effects of additional software requirements (Shahid et al., 2011).

In addition, the tool is capable of enforcing the creation and modification of a specific traceability link. If any alteration occurs on a requirement, the traceability information

may remain constant. Furthermore, the tool reads requirements, and then imports them from requirement specification documents generated outside the tool, while supporting the generation of descriptive documents for the requirements within the tool (Shahid et al., 2011).

#### **4.3.3 Comparison between RT practices**

Systematic review includes a highlight on the practices that are used in RT, due to its great importance in the requirements management (Zhou, 2014), controlled requirements, increased product quality, and monitoring the changes in requirements during SDLC (Kirova et al., 2008). The findings of the systematic review reveals the existence of a set of practices that are used to trace requirements, where the general trend of these practices are either automatic or semi-automatic rather than manual methods. They are beneficial in order to reduce the efforts in creating, maintaining and updating links and relationships of traceability (Huang, Berenbach, & Clark, 2007), reduce the time and efforts required to follow the requirements (De Lucia, Fasano, & Oliveto, 2008; Egyed et al., 2009), overrun the mistakes that occur if tracing manually (Cuddeback et al., 2010), as well as difficulties of traceability in large projects that contain a large number of requirements (Cleland-Huang et al., 2011).

There are varying levels of automation between the traceability techniques and tools. In regards to this, this study found three main levels of the automation; manual, semi-automatic, and automatic. Manual level refers to practices that require human efforts to establish and maintain traceability relations. Meanwhile, semi-automatic level refers to automatic methods that still need human activities to monitor the results produced by automated methods and to provide feedback and retrace. In contrast, automated level refers to practices that generate traceability relations as a result of the software development process.



it used to model system functionality and to generate functional test cases (Egyed & Grunbacher, 2002).

With regard to EB and GC, the main purpose of EB is to provide maintenance of traceability relationships (Cleland-Huang et al., 2002), while GC facilitates the management of the impact of functional change upon the non-functional requirements (Cleland-Huang et al., 2005). traceMAINTAINER is used to maintain post-requirements traceability amongst the elements of structural UML models (Mäder et al., 2009b). ArchTrace is a technique used to support the advancement of traceability links between architecture and implementation (Murta, Van Der Hoek, & Werner, 2006). Lastly, MB helps organizations and companies to get complete advantage from the traces constructed and to permit the project stakeholders to plan and produce trace schemes, and to execute them in a modeling environment which is in a graphical form (Cleland-Huang, Hayes & Domel, 2009).

On the basis of the extracted information from the systematic review and as discussed in this section, the techniques can be classified according to the type of requirements that are tracked: functional requirements, non-functional requirements and other both. Particularly, VB is used to trace the functional requirement Heindl and Biffl (2005), while ArchTrace and GC are used to trace the non-functional requirements (Murta, van der Hoek, & Werner, 2008; Cleland-Huang et al., 2005). As for the rest of the techniques, they trace both functional and non-functional requirements. Table 4.1 illustrates factors that help software companies to select the appropriate practices in their software development.

The functions of requirement traceability practices are the creation, maintenance, and update of traceability links (traceability relations) between artifacts such as requirement, test cases, source code, design documents, etc. Creation of traceability links is the act of

creating or defining traceability relations between artifacts to keep track of links, and to establish knowledge behind the existence of links. Some changes occur on traceability links while undergoing product development processes such as deleting or adding a link, known as maintenance traceability link. Where only edit processes are applied on a traceability link without deleting or adding, such processes are known as updates. The following table presents a set of RT practices used for these processes.





Table 4.1

*Factors that help software companies to select the appropriate RT practices (For references, please refer to Appendix F).*

RT Practices	Automation			Type of Req.		Traceability links			Size of Project			Benefits	Conclusions
	Automatic	Semi-Automatic	Manual	Functional	Non-Functional	Creation	Maintenance	Update	Small	Median	Large		
Rule-Based (RB)	√			√	√	√	√				√	<ul style="list-style-type: none"> <li>• Reduce effort.</li> <li>• Support full generation for artifacts evolution.</li> </ul>	<ul style="list-style-type: none"> <li>• The main aim of the RB technique is to automatically generate RT links using rules and maintenance traceability link.</li> <li>• The rules used in this technique are IREQ and RTOM.</li> <li>• Rules in this technique is used on the following documents; requirement statement document, analysis object models and use case documents.</li> </ul>
Value-Based (VB)		√		√		√	√				√	<ul style="list-style-type: none"> <li>• Reduce time.</li> <li>• Reduce effort.</li> <li>• Focus on the important requirements to trace.</li> <li>• Economic technique.</li> </ul>	<ul style="list-style-type: none"> <li>• The traceability efforts are reduced by focusing on most important requirements as compared to full tracing.</li> <li>• The prioritization step of VB identifies important requirements to be traced in more detail than others.</li> <li>• In VB, important requirements are identified</li> </ul>

RT Practices	Automation			Type of Req.		Traceability links			Size of Project			Benefits	Conclusions
	Automatic	Semi-Automatic	Manual	Functional	Non-Functional	Creation	Maintenance	Update	Small	Median	Large		
													based on parameters stakeholder value, requirements risk/volatility and tracing costs.
Information Retrieval (IR)	√			√	√	√					√	<ul style="list-style-type: none"> <li>• Reduce time.</li> <li>• Reduce effort.</li> <li>• based on natural language text.</li> <li>• Can reviewing each RT link in order to accept or reject.</li> </ul>	<ul style="list-style-type: none"> <li>• It is very hard to maintain links in constantly evolving systems (IR methods facilitate dynamic link generation).</li> <li>• IR provides a practicable solution to the problem of semi-automatically recovering traceability links between code and documentation</li> </ul>
Scenario-Based (SB)		√		√	√	√					√	<ul style="list-style-type: none"> <li>• Can help to minimize ambiguity in manual links.</li> <li>• Reduce time.</li> <li>• Reduce effort.</li> <li>• A feasible solution for requirements modification management activities.</li> <li>• Impact analysis of changing requirements.</li> </ul>	<ul style="list-style-type: none"> <li>• Scenarios are used to model system functionality and to generate functional test cases.</li> <li>• Scenario-based test cases create a mapping between requirements and other artifacts like design and code.</li> <li>• SB can help to minimize ambiguity in manual links by establishing dynamic verification in large systems.</li> <li>• SB is a feasible solution for requirements modification management activities in addition</li> </ul>

RT Practices	Automation			Type of Req.		Traceability links			Size of Project			Benefits	Conclusions
	Automatic	Semi-Automatic	Manual	Functional	Non-Functional	Creation	Maintenance	Update	Small	Median	Large		
													to impact analysis of changing requirements.
Event-Based (EB)		√		√	√		√			√		<ul style="list-style-type: none"> <li>• When modifications occur, it can be efficiently disseminated throughout the system of artifacts.</li> <li>• Helps developers to understand change effects.</li> <li>• Enables project managers to decide on whether new changes should be implemented or not.</li> </ul>	<ul style="list-style-type: none"> <li>• The basic purpose of the RB technique is to manage and maintain traceability relationships between requirements and UML along with test cases.</li> <li>• In EB, when change occurs in requirements, an event message is published, which is then notified to all dependent objects.</li> </ul>
Hypertext-Based (HTB)		√		√	√	√				√		<ul style="list-style-type: none"> <li>• Reduce effort.</li> <li>• Tracing between different artifacts.</li> <li>• Reduce time.</li> </ul>	<ul style="list-style-type: none"> <li>• The HTB technique is used for traceability between different artefacts.</li> <li>• HTB is a technique used for traceability link generation using open hypermedia and</li> </ul>

RT Practices	Automation			Type of Req.		Traceability links			Size of Project			Benefits	Conclusions
	Automatic	Semi-Automatic	Manual	Functional	Non-Functional	Creation	Maintenance	Update	Small	Median	Large		
													information integration.
ArchTrace		√			√		√	√		√		<ul style="list-style-type: none"> <li>• Support for versioning the relationships between architecture and implementation as the architecture or code evolves with time.</li> <li>• Provides a policy-based infrastructural to perform automatic update.</li> <li>• Can add a new policies.</li> </ul>	<ul style="list-style-type: none"> <li>• The main objective of ArchTrace technique is to the advancement of traceability relationships between the architecture and implementation.</li> <li>• ArchTrace differentiates itself by updating traceability links from architectural elements to code elements.</li> <li>• This technique permits developers choose a set of traceability management policies that is suitable for their situational needs and working styles.</li> </ul>
Model-Based (MB)		√		√	√	√	√			√		<ul style="list-style-type: none"> <li>• Reduce cost of creating and maintenance of RT links.</li> <li>• Obtain complete advantage from the traces constructed.</li> </ul>	<ul style="list-style-type: none"> <li>• The purpose of the MB technique is to determine artifact granularity suitable for traceability and to extract all candidate traceability links in order to achieve a recall rate of 100%.</li> </ul>

RT Practices	Automation			Type of Req.		Traceability links			Size of Project			Benefits	Conclusions
	Automatic	Semi-Automatic	Manual	Functional	Non-Functional	Creation	Maintenance	Update	Small	Median	Large		
												<ul style="list-style-type: none"> <li>• Help to plan and produce trace schemes.</li> </ul>	<ul style="list-style-type: none"> <li>• To assist companies obtain complete advantage from the traces constructed and to permit the project stakeholders to plan and produce trace schemes, to execute them in a modeling environment which is in a graphical form.</li> </ul>
Goal-Centric (GC)		√			√		√	√		√		<ul style="list-style-type: none"> <li>• Recognizing the initial impact of a modification</li> <li>• Increase a quality of system.</li> <li>• Deliver greater degrees of automation for utilizing and comprehending traceability links.</li> </ul>	<ul style="list-style-type: none"> <li>• GC facilitates to manage impact of functional change upon the non-functional requirements.</li> <li>• GC helps to manage critical system qualities such as safety, security, reliability, usability and performance.</li> </ul>
RTM			√	√	√	√	√	√	√			<ul style="list-style-type: none"> <li>• Easy to use and apply.</li> <li>• Easy to understand.</li> <li>• Used to manage modifications.</li> <li>• Used as base for test</li> </ul>	<ul style="list-style-type: none"> <li>• RTM is used to identify the traceability relationships between requirements and other kinds of artifacts.</li> <li>• It is utilized in the management of modifications</li> </ul>

RT Practices	Automation			Type of Req.		Traceability links			Size of Project			Benefits	Conclusions
	Automatic	Semi-Automatic	Manual	Functional	Non-Functional	Creation	Maintenance	Update	Small	Median	Large		
												planning. • Provides ease in track changes.	and as the base for test planning. • Preparing traceability matrix ensures that every functionalities required of the application in the test cases has been covered. • RTM needs to a long time to create and prone to mistakes because it manual method.
traceMAINTAINER		√		√	√		√	√		√		• Reduce effort. • Reduce time.	• traceMAINTAINER is used to update of traceability relationships between analysis models, requirements, and design models of the software systems which are expressed in UML. • traceMAINTAINER allows the update of traceability relations by developers, with little manual effort.
Retro	√			√	√	√					√	• Reduce effort. • Reduce time. • To overcome the manual traceability problems.	• RETRO is used to ease the automatic creation of requirements traceability matrices (RTM). • RETRO utilizes IR and text mining methods to construct candidate traces. • Create traceability links more accurate than

RT Practices	Automation			Type of Req.		Traceability links			Size of Project			Benefits	Conclusions
	Automatic	Semi-Automatic	Manual	Functional	Non-Functional	Creation	Maintenance	Update	Small	Median	Large		
												<ul style="list-style-type: none"> <li>• Easily create traceability matrix.</li> </ul>	manual methods and used only one-third of the usual time to do so.
DevComplete				√	√	√	√	√	–	–	–	<ul style="list-style-type: none"> <li>• Provide full traceability.</li> <li>• Capable of enforcing the creation and modification of a specific traceability link.</li> </ul>	<ul style="list-style-type: none"> <li>• It offers a complete traceability for project tasks, requirements tracking, and flaws for team agility improvement.</li> <li>• DevComplete permits freedom to implement requirement modifications according to current demands.</li> </ul>
DesignTrack				√	√	√	√		–	–	–	<ul style="list-style-type: none"> <li>• Can be utilized in the management of design requirements issues.</li> <li>• Offers an environment for the navigation of intricate design information spaces through supporting RT.</li> </ul>	<ul style="list-style-type: none"> <li>• It provides traceability between requirements and architectural design.</li> <li>• This tool also provide an integrated environment for designers to manage requirements information along with exploration.</li> </ul>

Indeed, systematic literature review showed that information retrieval is a technique the most commonly used among RT practices to trace requirements. The selected studies in SLR shows that 15 studies are used or contains information about information retrieval. Indeed, this is due to several reasons; information retrieval is an automatic method to create traceability links between two types of artifacts, another big advantage is that used to trace the functional and non-functional requirements, and additional advantages are that information retrieval reduces the time and effort to create traceability links between artifacts. IR technique have been implemented on RETRO tools (Pandanaoyana et al., 2013). Table 4.2 illustrates the traceability techniques that used with traceability tools.

Table 4.2

*Matching between the tools and techniques of traceability and the tools price*

RT Tool	RT Technique	Price/Cost	References
<b>RTM</b>	It is a manual tool that can be applied on form of a table or any of spreadsheet programs (such as Microsoft office excel).	Free	(Athira & Samuel, 2011)
<b>traceMAINTAINER</b>	It is a tool that supports a techniques for post-requirement traceability relations. As result of this, this tool supports all the techniques mentioned in this study.	N/A	(Mäder et al., 2009b)



<b>RETRO</b>	Information Retrieval (IR)	Commercial	(Cuddeback, Dekhtyar, & Hayes, 2010; Pandanaboyana et al., 2013; Shahid, Ibrahim, & Mahrin, 2011)
<b>DevComplete</b>	The literature did not identify technique used with this tool.	Commercial	-
<b>DesignTrack</b>	This tool has not been evaluated experimentally yet.	N/A	(Shahid, Ibrahim, & Mahrin, 2011)

#### 4.4 Discussion

The concept of RT is a mechanism which serves as a guide for the auditing of traceability relationships or links between requirements and artifacts. It is essential for software development because a great amount of information is produced and used, which should be kept related (traceable). When systems are being developed (especially large systems), it is difficult or impossible to recall all links that were made to relate information. For this reason, RT represents an important stage in development projects (Pinheiro, 2004).

Some advantages of RT can be discerned. Firstly, traceability provides requirements verification. During this process, a comparison is made between the requirements specification and the system to verify the fulfillment of demands (Blaauboer, 2006). Secondly, a software which has undergone complete traceability allows for a lot easier analysis of the impact of modifications or changes. The consequences or results of a modification can be traced from requirement to design, code and test cases, which

significantly shortens and improves the process (Arkley & Riddle, 2005). Changes occur during software development, mostly in the maintenance stage. The effort needed for the preservation of this change, depends on the complexity and size of its environment. The use of traceability in minimizing this complexity can mitigate costs as a result of impact analysis. This (impact analysis) is often easier than the entire change effort (Blaauboer, 2006; Niessink & Vliet, 1998).

Thirdly, RT is a criterion for consistent change integration and effective system maintenance. In addition, the use of traceability in software systems results in an increase in software qualities, whereas overlooking or not using traceability brings about negative effects to a project. An instance of such negative effects is a decrease in system quality which results in revisions, and hence, increases project time and cost (Morckos, 2011).

Finally, traceability ensures customer satisfaction which is one of the main benefits acquired from it. Traceability enables customers to follow up and monitor the software development process according to their requirement. RT offers a guarantee that all requirements are implemented, which benefits product managers and requirements engineers (Torkar et al., 2012).

Indeed, the main purpose of this study is to explore the RT practices that assist software companies to trace requirements during software development process and also help companies in determining the appropriate method for their work by identifying a set of factors of each practice.

As aforementioned, this study explored the most traceability practices adopted and used by previous related works. The researcher conducted intensive literature review (Systemic literature review). According to Ramesh and Jarke (2001), every traceability link costs money to create and maintain, and it has also been determined that not all

requirements are equally critical and not all should be traced. In fact, systematic review presented a technique called the value-based (VB) technique. This technique can determine requirements that are very important to trace, and also those that are less valuable to trace. Thus, it provides an economic and a technical model to software companies for tracing requirements.

Cleland-Huang et al., (2011) explained that creating a traceability link and maintaining it between different artifacts, takes a lot of time and effort in addition to it being prone to errors. Their study provides a technique for the purpose of reducing the time and effort utilized in the creation and maintenance of traceability links. The technique was termed the event-based (EB) technique. It creates links between software artifacts after a change request is executed, and alleviates the coordination efforts required for maintaining software artifacts. In other words, the technique does not create a traceability link until a change occurs in the requirements such as refinement, replacement, abandonment, or merge. In addition, the study also reviewed two techniques; information retrieval (IR) and rule based (RB) techniques. The IR technique concentrates on automating the creation of traceability links using comparison based on similarity between two types of artifacts. This serves to reduce the time and effort spent on practitioners because it works automatically. In the same vein, traceability links are generated automatically using the rule-based (RB) technique. The RB technique does not support partial generation for artifacts evolution. If change has been applied to a certain artifact, then traces will be re-generated by the RB technique as it produces links for the first time.

The studies by Cysneiros, (2007) and Mahmoud (2015) discussed that NFRs are hard to trace along the different phases of the software development process. Similarly, the rationale on how to cope with the needs of NFRs, is hard to trace. Indeed, the findings

from the systematic review revealed that there are two practices or methods used to trace non-functional requirements. These practices are called ArchTrace and Goal-Centric. Goal-Centric provides traceability anchorage for maintaining and managing (NFRs), and associated quality concerns through the long life span of a software intensive system. As for the ArchTrace, it provides a tool support for versioning the relationships between a component-and-connector architecture, and its execution as the architecture or code evolves with time. Both of these practices are used to trace non-functional requirements only. Additionally, systematic review also provides a set of RT practices are used to trace functional and non-functional requirements such as RB, IR, SB, EB, MB and HTB.

With regards to traceability tools, one of the challenges facing the execution and application of traceability requirements in project development is the lack of sufficient tools for this purpose (Aizenbud-Reshef et al, 2006; Schwarz, Ebert, & Winter, 2010). This study also provides a group of traceability tools that assist in the implementation of requirement traceability in project development such as RTM, Retro, traceMAINTAINER, DevComplete, and DesignTrack. The RTM is mostly utilized in industries to determine the relationships between requirements and other kinds of artifacts such as test cases, code modules, and design. It is also used in defining the relationship between requirements and other artifacts such as design modules, code modules, and test cases. Another use of this tool is in the management of modifications and as the base for test planning. Preparing traceability matrix ensures that every functionalities required of the application in the test cases has been covered. Although traceability matrix is easy to understand by practitioners, it is a manual tool that requires time and effort to create.

Requirements tracing by manual traceability methods are prone to errors and also time-consuming (Cuddeback et al., 2010). Systematic review includes a tool called Retro, which is a traceability tool for the automated creation of traceability matrix. Thus, it facilitates the production of traceability matrix and does not require the time and effort spent in the case of manual methods. The traceMAINTAINER is another tool which anchors the semi-automated update of traceability connections between analysis, requirements, and design models of the software systems expressed in UML. This tool allows for the update of traceability relations by developers, with little manual effort.

According to Bouillon et al., (2013), the usage of RT is less common in development projects. Therefore, systematic review provides a set of traceability practices that are used for different purposes such as creating, maintaining, updating, and evaluating of traceability links during the stages of project development.

This study also reviewed two tools for the purpose of traceability; the DevComplete and the DesignTrack. The DesignTrack offers an environment for the navigation of intricate design information spaces through supporting requirement traceability (Ozkaya & Akin, 2007). It also calculates the power, applicability, and the limitations of computer-aided designs which are enabled by requirement traceability. In addition, it can be utilized in the management of design requirements issues. The DevComplete on the other hand, offers a complete traceability for project tasks, requirements tracking, and flaws detection for team agility improvement. This tool permits freedom to implement requirement modifications according to current demands. It also detects the effects of additional software requirement (Shahid et al., 2011).

#### **4.5 Summary of Chapter**

This chapter presents a systematic literature review of requirements traceability practices over the period from 2008 to 2015. Results shows a set of techniques and tools

used to trace requirements during the software development life cycle. It also identifies the purpose of each method and its use in tracing any kind of requirements.



## **CHAPTER FIVE**

### **RESULTS OF SURVEY AND INTERVIEW**

#### **5.1 Introduction**

This chapter elaborates the data obtained from the respondents. It reflects the use of the software companies to requirements traceability practices in their software development tasks. Further, the results of interviews with some of the practitioners about the use of traceability practices in the software companies.

#### **5.2 Results of the Use of Traceability Practices**

The questionnaire contains three sections. First, it asks about the demographic details of the participants. Then, it asks about the use of RT tools, which is followed with the use of RT techniques. Altogether, 31 software companies located in the northern region of Malaysia participate in this study. It is sufficient for this study, as recommended by Sekaran and Bougie (2010). Engaging software companies is a tough job. Various techniques were used for that, including phone calls, email, and personal networking through snowballing technique.

##### **5.2.1 Demographic Characteristics of Respondents**

The respondents participating in this study consist of software developers, requirement analysts, project managers, designers, and testers who work in software companies in the northern region of Malaysia. They have been very experienced in their field. The distribution is diverse, as detailed in Table 5.1.

Table 5.1

*Distribution of respondents*

No	Demographic profile	Frequency	Percentage
1	<b>Gender</b>		
	Male	11	35.5
	Female	20	64.5
2	<b>Age</b>		
	18-24	1	3.2
	25-34	23	74.2
	35-44	6	19.4
	45-50	1	3.2
3	<b>Education Level</b>		
	Bachelor degree	23	74.2
	Master degree	7	22.6
	Other	1	3.2
4	<b>Organization Level (team role)</b>		
	Project manager	2	6.5
	Requirement analyst	5	16.1
	Designer	3	9.7
	Developer	19	61.3
	Tester	1	3.2
	Other	1	3.2
5	<b>Sector of the Software Company</b>		
	Communication	3	9.7
	Financial	1	3.2
	Commercial	14	45.2
	Other	13	41.9
6	<b>Years of Experience</b>		
	Less than 5 years	11	35.5
	5-10 years	16	51.6
	More than 10 years	4	12.9
7	<b>No. of Employees in the Company</b>		
	5-10 people	6	19.4
	10-50 people	14	45.2
	More than 50 people	11	35.5
8	<b>Number of Software Projects involved in</b>		
	Less than 10 projects	14	45.2
	10-20 projects	14	45.2
	More than 20 projects	3	9.7



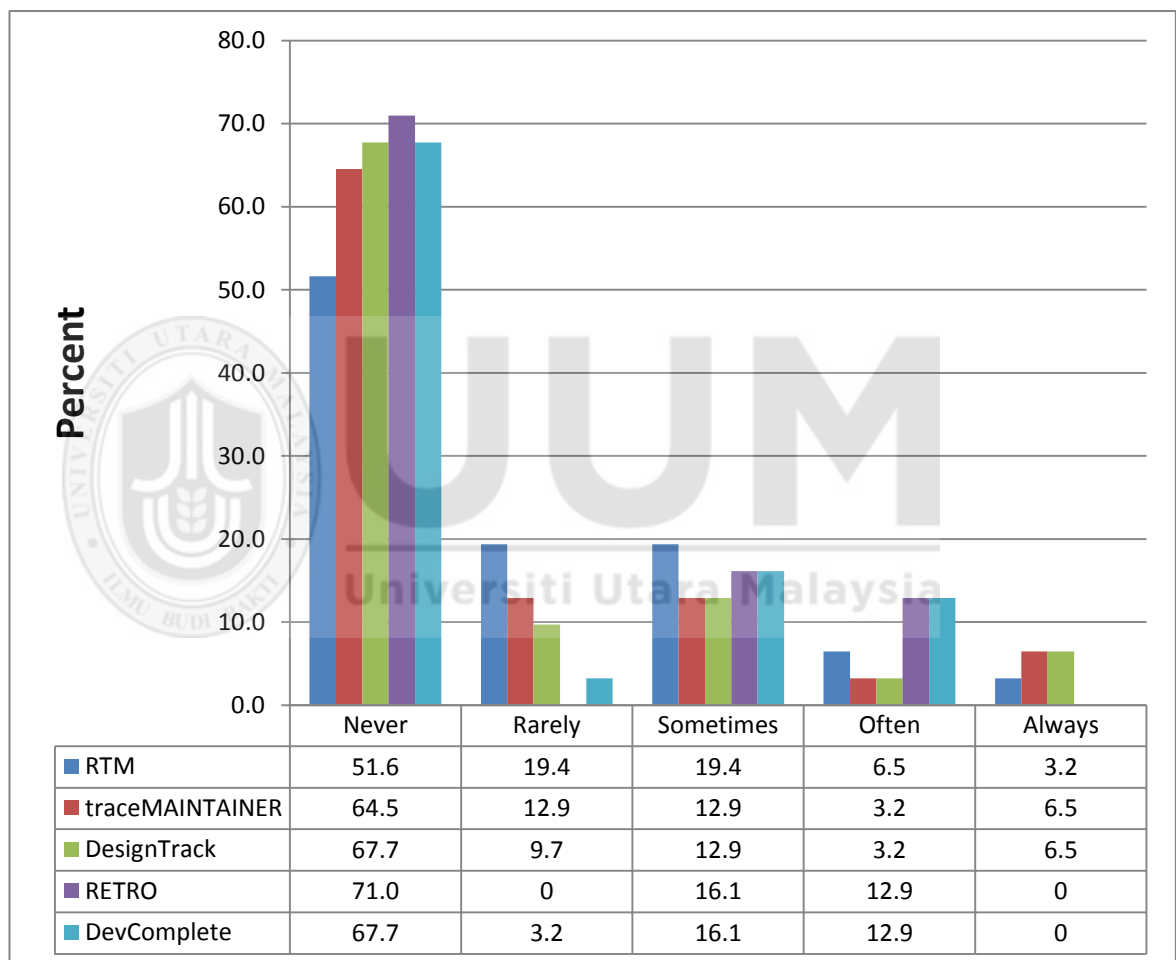
Table 5.1 shows that out of the 31 respondents, 11 (35.5%) of them are male, while 20 (64.5%) are female. Most of them (23 (74.2%)) are between 25 and 34 years. The rest are between 35 and 44 years old (6 (19.4%)), one person (3.2%) between 18 and 24 years old, and one person (3.2%) is between 44 and 50 years. Most of them (74.2%) have a bachelor degree. Besides, 22.6% have a master degree, while 3.2% have another certificates. Since most of them have a bachelor degree, majority of them are developers (61.3%). However, 16.1% are requirements analysts, which is quite high. Besides, there are also designers (9.7%), project managers (6.5), and testers and other roles (3.2% respectively). The respondents represent companies in various business sectors. Most of them (14 (45.2%)) are in commercial software sector. Not many of the companies are venturing in communication (9.7%) and finance (3.2%), because many of them (13 (41.9%)) are venturing in various other sectors.

Among the participants, 45.2% of them work in medium-sized companies (between 10 and 50 employees). Mostly (35.5%) work in large companies, with more than 50 employees. Meanwhile, the rest 19.4% work in small companies with 5 to 10 employees. In the companies where they are currently serving, more than half of them (51.6%) have been in there between 5 and 10 years. Majority of them (35.5%) are still new with the companies (less than 5 years), while the rest (12.9%) have already served for more than 10 years.

Most of the participants have involved in not more than 20 projects. In detail, 45.2% of them have participated in less than 10 software projects and another 45.2% between 10 and 20 projects. The other 9.7% have participated in more than 20 projects.

### 5.2.2 Use of Requirements Traceability Tools

From the analysis of the requirements traceability tools, the rates of use of each tool are grouped into ‘Never’, ‘Rarely’, ‘Sometimes’, ‘Often’, and ‘Always’, as illustrated in Figure 5.1. Five (5) tools were analyzed namely RTM, traceMAINTAINER, DesignTrack, Retro, and DevComplete. Referring to Figure 5.1, it is clear that the trend for all five tools is similar. Basically, most of the participants have never used the tools.



*Figure 5.1: Use of Traceability Tools in Software Companies*

In detail, it is seen that 51.6% have never used RTM for tracking requirements. Meanwhile, 19.4% of them have rarely or sometimes used it. However, 6.5% of them have used it often, and 3.2% has always used it in tracking requirements. For traceMAINTAINER, 64.5% of the participants have never used, while the other 12.9%

have either rarely or sometimes used. But, it has often been used by 3.2%, and always used by 6.5% of the participants. The percentage of last three categories are shared with DesignTrack. However, there are more participants (67.7%) never used DesignTrack than that for traceMAINTAINER, leaving only 9.7% rarely used.

RETRO and DevComplete have often been used by 12.9% of the participants, but no one has always used them. The other 16.1% have sometimes used them, while 71% have never used RETRO. It is different with DevComplete, where 67.7% have never used, while the other 3.2% have rarely used.

### **5.2.3 Use of Requirements Traceability Techniques**

Similar to the analysis of the requirements traceability tools, the rate of use were grouped into 'Never', 'Rarely', 'Sometimes', 'Often', and 'Always', for the analysis of the requirements traceability techniques (refer to Figure 5.2). Nine techniques were analyzed namely EB, IR, RB, ArchTrace, VB, SB, HTB, GC, and MB. Similar with the findings on the tools, most of the participants have never used the techniques. Only few of them have either often or always used them. This is detailed in the subsequent paragraphs.

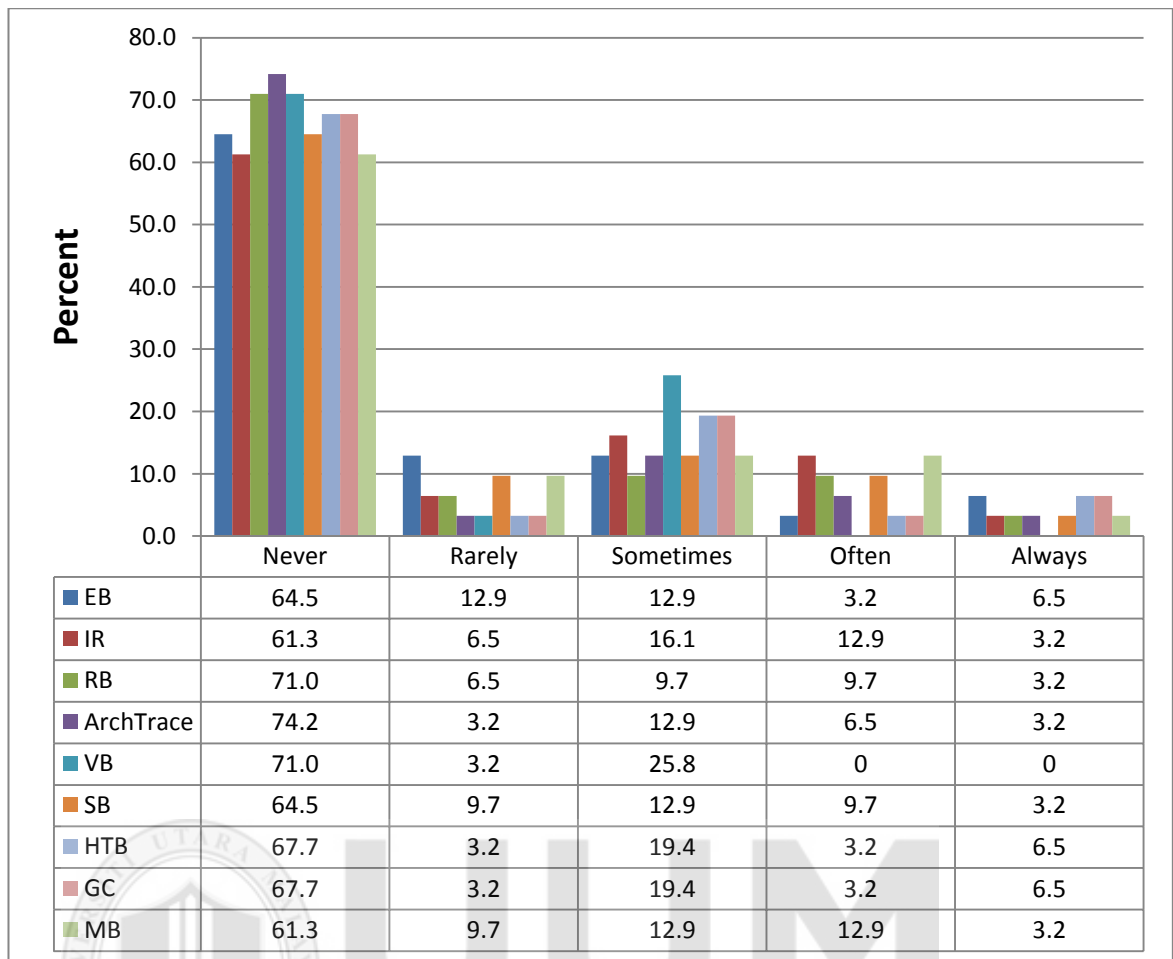


Figure 5.2: Use of Traceability Techniques in Software Companies

Figure 5.2 exhibits that between 61.3% and 74.2% of the participants have never used the techniques. In contrast, only between 3.2% and 6.5% of the participants have always used them, and between 3.2% and 12.9% have often used. The rest have either rarely used (between 3.2% and 12.9%) or sometimes used (between 9.7% and 25.8%).

In detail, ArchTrace has never been used by 74.2% of the participants, the highest. At the same time, it has always been used by 3.2% of the participants, the lowest. In contrast, IR and MB have often and always been used by 12.9 % and 3.2% of the participants. At the same time, only 61.3% of the participants have never used them. Between these two, IR is better because only 6.5% of the participants rarely used it, compared to MB (9.7%). VB is the worst because no participant have either often or always used.

### 5.3 Reliability For Use of Traceability Practices

Reliability is the amount of which an experiment, test or even measurement process, is expected to yield the same outcome on a recurrent trial (Sekaran & Bougie, 2010). According to Zikmund (2003), reliability simply means the extent to which measurement tools are free from error and, therefore, produce a consistent result. Based on that, this study has carried out reliability tests, to ensure results are free of doubt. The result of the test is exhibited in Table 5.2.

Table 5.2

#### *Reliability Result*

Cronbach's Alpha	N of Items
0.987	14

Regarding the reliability, Sekaran and Bougie (2010) underline that alpha values less than 0.60 are considered poor, while those between 0.6 and 0.8 are acceptable. Further, values greater than 0.8 are considered good. Based on such classification, with reference to Table 5.2, the alpha value for this study (0.987) is very high. This explains that the tool is reliable in gathering data it should gather, and that the results are free of doubt.

### 5.4 Descriptive Statistics

Table 5.3 details the results on the use of traceability practices in software companies during software development. They answer 14 items with answers from 1 (never) to 5 (always). Referring to the table, the mean for all items is less than 2. This explains that the participants have never or rarely used it during software development.

Table 5.3

*Descriptive Statistics*

	<b>N</b>	<b>Minimum</b>	<b>Maximum</b>	<b>Mean</b>	<b>Std. Deviation</b>
<b>Q1</b>	31	1	5	1.90	1.136
<b>Q2</b>	31	1	5	1.74	1.210
<b>Q3</b>	31	1	5	1.71	1.216
<b>Q4</b>	31	1	4	1.71	1.160
<b>Q5</b>	31	1	4	1.74	1.154
<b>Q6</b>	31	1	5	1.74	1.210
<b>Q7</b>	31	1	5	1.90	1.274
<b>Q8</b>	31	1	5	1.68	1.194
<b>Q9</b>	31	1	5	1.61	1.145
<b>Q10</b>	31	1	3	1.55	.888
<b>Q11</b>	31	1	5	1.77	1.203
<b>Q12</b>	31	1	4	1.58	.923
<b>Q13</b>	31	1	5	1.77	1.257
<b>Q14</b>	31	1	5	1.87	1.258
<b>Valid N (listwise)</b>	31				

**5.5 Analysis of Interviews**

Having analyzed the feedback in the questionnaire, the norm is clear that the participants never use traceability practice in developing software. It is an interesting finding, and invokes interest to discover richer information, especially the reasons leading to it. Accordingly, this study carried out an interview with two persons who have good experience with software development, particularly in Malaysia companies. Since interview is a technique that could discover rich information, involving two subjects is enough, as recommended by Clark and Creswell (2014). Interviews with practitioners have been recorded. The demographic details of the participants in the interviews are provided in Table 5.4.

Table 5.4

*Participant's Information*

No.	Property	Participant 1	Participant 2
1	Gender	Male	Female
2	Qualification	Master degree	Master degree
3	Experience	More than 4 years	More than 4 years
4	Org. level (team role)	Software Developer	Software Developer
5	No. of Projects involved	More than 10 projects	More than 10 projects

Indeed, the interview session includes several open-ended questions pertaining to RT practices. The first question posted to the interviewees was aimed at clarifying whether their companies use RT practices or not in their software development process. Both of the participants stated that most of the software companies do not implement any of the RT practices throughout the software lifecycle. With regard to why the companies do not apply such practices through the work process, Participant 1 indicates that:

*"I think the main reason is the lack of knowledge on these practices, as well as the unclear benefits of them and their significance. Additionally, the cost for adopting the practices in the software development process also determines".*

Similarly, Participant 2 highlight:

*"This is due to several reasons such as the companies do not know the benefits of RT practices, or there is no knowledge about traceability practices. Also, the cost may be one of the reasons".*

Their answers reflect that lack of knowledge on RT practices and its benefits is considered as an important reason that discards the implement of its practice in software

companies. Nevertheless, the financial and training constraints, as well as personal experience are the barriers to implement the RT practices. Regarding to the implementation of RT practices, this study raised a question on whether RT practice will help software companies produce high-quality software. The answer from Participant 1 sounds:

*"...it helps to manage requirements and their effects, and know that requirements gathering is very important. So, if the use of this practice helps maintaining requirements and follow their changes, therefore, it will help to present a good software".*

Participant 2 agrees by adding something when saying:

*"..used of techniques and tools to follow requirements will greatly facilitate the software development process, because this method will ensure all requirements gathered from stakeholders or customers will appear in the final product, and any change in requirements during the design of the software will be easily detected".*

In general, the purpose of using these practices help companies to trace requirements, maintain, and update changes. Thus, this ensures no loss of requirements and ensures their presence in the final product. As a result of implementing this practice will lead to high quality software that embed all requirements and characteristics that have been extracted during requirements development phase. In terms of the kinds of requirements that are most preferred by software companies to apply RT practices, Participant 1 responded:



*"Both the functional and non-functional requirements are important to trace and must be preserved and maintained during the software development process. This is an essential task, especially in large projects that consist of a large number of requirements".*

Participant 2 confirmed what participant 1 said, by saying:

*"All kinds of requirements are necessary for the success of the project requirements. With this, traceability must be applied on both".*

Both participants are aware that the main reasons for the success of any project is to keep and maintain requirements during the product development. Tracing functional and non-functional requirements is necessary to increase product quality and prevent re-work. As for the software development stage that requires extensive traceability application, Participant 1 replies:

*"The requirements evolve and undergo many changes during the development process. Therefore, maintenance is one of the most important stages that require the application of traceability after every change in requirements".*

Participant 2 believes that traceability in all stages of software development is required, in his words:

*" Applying RT practices is important in managing requirements during all phases of project development ".*

It is understandable that applying traceability in all phases of the software development is important. Tracing requirements from the source to the final project is essential to

ensure the success of the project, by maintaining the existence of requirements without change in the final product. This is most important in the maintenance phase, in order to trace changes in the traceability links, and maintain it.

Lastly, the participants were asked on their recommendations for companies that still do not use the RT practice, and how they can be applied. Participant 1 suggested that:

*"If the traceability practices are very beneficial, I think we need to expose them about these practices. When they know the importance of these practices, then they will be interested to apply the techniques in their software development".*

Participant 2 adds:

*"Depending on the benefits of those techniques and tools, it will be easy to follow requirements that will later become system specifications, my suggestion for companies to consider these practices because the requirements gathering process and control takes a long time and effort. The practices help make the process much easier"*

As a recommendation, companies must consider the practices for their benefits, which is also significant in preventing failures in the program. Besides, it helps saving time and effort in following the changes in the requirements that occur during the software development life cycle. All these benefits help in managing requirements and that they can be easily controlled and thus facilitate software development process.

Based on the answers also, it is noticed that both of the participants realize the importance of the RT practices in supporting the development efforts. However, in common situation, due to the deadline stated by the users to deliver the software, the practitioners do not have enough time to learn or apply RT practices. In addition,

software companies need to invest a suitable practice to support the software development task. When these could be done, it could lead to a high quality product that meets all the necessary requirements for the end user.

## **5.6 Discussion**

This empirical study provides evidence that there is insufficient use of requirement traceability practices during projects development in software companies. Also, it clarified and discovered the reasons why most software companies in Malaysia pay less attention to traceability requirement.

To investigate the current RT practices applied among software companies in Malaysia, the study distributed a series of questionnaires to a set of software companies located in the northern region of Malaysia. It was strategized and accelerated to discover the traceability practices used by those companies. The questionnaires were distributed by email and the period of data collection was approximately thirty days. The results revealed that majority of software companies do not use RT practices in project development processes. Consequently, this study provides empirical evidence on the usage of traceability practices during project development in software companies.

Finally, the study aims at finding out the barriers that prevent software companies or practitioners in Malaysia who work in it from applying RT practices. A series of interviews were conducted with experts in the field to find out the reasons. It was discovered that the reasons include lack of knowledge or the prevalence of these practices, as well as financial issues for purchasing or training for the purpose of RT practices. These discoveries (results from this study), conforms to the results of previous studies (Cleland-Huang, Chang, & Christensen, 2003; Jaber, Sharif, & Liu, 2013; Winkler & Pilgrim, 2010).

## 5.7 Summary of Chapter

This chapter explains the findings of this study through a survey on the use of requirements traceability practices among software companies. Also, an interview was conducted with two practitioners to find out the reasons behind the lack of application of these practices in the companies and their recommendations for applying of these practices. Further, the next chapter elaborates the contributions, limitations of this study, and recommendations for future work.



## **CHAPTER SIX**

### **CONCLUSION AND RECOMMENDATIONS**

#### **6.1 Introduction**

This study focuses on the practices of RT as an important and decisive part in software development. This chapter discusses the results that have been obtained through a systematic literature review, the results of the questionnaire and interview, as well as the achievement of objectives stated in Chapter 1. Also, this chapter highlights the contributions and limitations of this study. Finally, the chapter concludes the work being carried out and recommends some future works to enhance this study.

#### **6.2 Objectives Achievement**

Having performed the activities outlined in Chapter 3 and discussed in Chapters 4 and 5, the study has achieved all the stated objectives in Chapter 1. In aggregation, they are discussed as follows:

**Objective 1** – To identify RT practices within software companies and factors that help these companies to select appropriate practices using systematic literature review.

The first objective of this study is to identify the practices of requirements traceability and a set of factors that assist software companies to select a suitable RT practices in their work. Depending on the protocol of systematic literature review in Chapter 3, a set of techniques and tools used in tracing requirements have been extracted from the literature (in works from 2008 to 2015). The traceability practices (techniques) that were extracted from systematic literature review are; Rule-Based, Value-Based, Information Retrieval, Scenario-Based, Event-Based, Hypertext-Based, ArchTrace, Model-Based, Goal-Centric. Whereas traceability practices (tools) are; Requirement

Traceability Matrix, traceMAINTAINER, Retro, DevComplete, and DesignTrack. Some of these practices are used in tracing the functional requirements of the system, and others are used in tracing non-functional requirements, while there are also the practices used for both. In addition, the practices are divided into three methods in order to trace requirements; manual, semi-automatic, and automatic. Furthermore, there are a variety of factors extracted through conducted a systematic literature review. These factors represent features of RT practices that enables software companies in determining which practices are appropriate for their work. These factors included automation of practices, type of requirement, processes of traceability links and size of projects. These results have been discussed in Chapter 4.

**Objective 2** – To investigate the current practices of requirement traceability applied among software companies in Malaysia quantitatively.

The second objective of this study is to investigate the current RT techniques and tools used in software companies. To achieve this objective, a set of questionnaire has been distributed to a set of software companies located in the northern region of Malaysia. It was strategized and accelerated to discover the traceability practices used by those companies. The questionnaire was distributed by email and the period of data collection were thirty days approximately. The results as discussed in Chapter 5 show that most companies do not use these practices when developing software. Where data collected from software companies indicated that around 64.5 percent have never used traceability tools, and around 67 percent have never used traceability techniques. The remaining of percentage is distributed between use rarely, sometimes, in addition to often and always. The findings point out that there is lack of interest in in the requirement management in general, and requirement traceability in particular.

**Objective 3** – To verify the requirement traceability practices used qualitatively with industrial experts.

The third objective of this study aimed at finding out the barriers that prevent companies from applying RT practices in addition to re-check and confirm the results of the second objective of this research. For that, this study conducted a series of interviews with practitioners who work in software companies and have experience in that domain. Interviews confirmed that the participants realize the importance of the RT practices in supporting the software development processes. The outcome of interviews illustrates that the reasons include lack of knowledge or the prevalence of these practices, as well as financial issues for purchasing and training.

### **6.3 Contribution of the Research**

This study has several theoretical implications and contributions to software engineering and requirement management. More precisely, this study includes valuable information about requirement traceability and practices used to it, because there is lack of studies in this area.

Firstly, this study has reviewed a set of RT practices that are used to trace requirements from their sources until the final product is completed. By using SLR, the results of this systematic review pointed to factors that can help software companies or practitioners to select the appropriate practices for their work and better understand of RT practices. These factors included automation of practices, type of requirement, processes of traceability links and size of projects. Factors have been discussed in Table 4.1. In the same vein, this study also highlights the various issues faced by the companies and organizations related to the requirements traceability aspect, such as time and effort required to create and maintain traceability link and find a suitable practices for that

purpose. Consequently, the outcome from this theoretical review may motivate new researchers to focus on these particular issues and thus assist the organizations or companies (practitioners in particular) to enhance their current software development processes.

Secondly, requirement traceability is very important and crucial for the success of software companies. But, unfortunately the prior studies give little attention to this phenomenon. Thus, this study provided knowledge in details about the RT practices which can be used for the software companies in various levels of traceability links such as create, update, maintain traceability links between requirements and different artifacts.

Thirdly, the practical side is very crucial to prove the theoretical arguments. However, there is paucity of the empirical evidence on the software companies related to requirement traceability. Therefore, this study also provides empirical evidence on the use of traceability practices through conducted survey within software companies in the northern part of Malaysia, which highlight the most practices used within these selected companies. Results of empirical evidence indicates that the majority of software companies do not use traceability practices in the development of software projects. As a result, there is a weakness in the tracing requirements in the projects in particular, in addition to lack of managing requirements in general.

Finally, in Malaysia particularly, there is no empirical evidence on the current RT practices used. Thus, the current study will add valuable of knowledge in this context. In addition, this study discovered reasons the lack of use RT practices among software companies in the software development processes, by conducted interviews with industrial experts of this field.



#### **6.4 Limitation of the Study**

The key purpose of this study to explore the main practices used among software companies. As of every study, this study has several limitations that should be noted. First, the respondents of this study are limited to software companies in the northern part of Malaysia only, thus limiting the generalization of the findings. Second, as any survey study there are several limitations due to the financial and unwilling from some of the companies to conduct the survey or the interview. This study carried out the survey only with thirty-one companies. As for the interview, only two experts accepted to be interviewed. Nevertheless, the results of this study could be generalized for software companies, because the majority of the companies included in this study are the branches for companies located in Kuala Lumpur and other major areas in Malaysia.

#### **6.5 Conclusion and Future Works**

This study could be improved in future research through more empirical research with different levels of projects development and increase sizes of respondents. It could also involve other parts in Malaysia through cross-country studies. In a nutshell, this study is conducted to identify the traceability practices based on the literatures as well as focus on factors which assist software companies to select the RT practices to be a suitable for their development processes. The discussion highlight the practices of requirements traceability, which affect the quality of the product effectively, control of requirements, and detection of changes that occur upon it. Also, this study conducted an empirical study to determine whether companies apply these practices or not, as well as the reasons that lead to the non-use of these practices in companies.

Overall, this study was carried out to investigate the current RT practices used by software companies, and extracting a set of techniques and tools that can assist them to

accomplish their task. More research in the RT area is required in order to enhance the understanding of the impact RT practices on software development. By doing this, it can help researchers and even software practitioners to plan and make right decision to select appropriate RT practices which can improve quality of software product.



## REFERENCES

- Aartsengel, A. Van, Kurtoglu, S., Lean, T., Sigma, S., & Methodology, S. (2013). *Handbook on Continuous Improvement Transformation*.  
<http://doi.org/10.1007/978-3-642-35901-9>
- Abbors, F., Truşcan, D., & Lilius, J. (2009). Tracing requirements in a model-based testing approach. In *Advances in System Testing and Validation Lifecycle, 2009. VALID'09. First International Conference on* (pp. 123–128). IEEE.
- Abdalla, M., Bourse, F., De Caro, A., & Pointcheval, D. (2015). Simple functional encryption schemes for inner products. In *Public-Key Cryptography--PKC 2015* (pp. 733–751). Springer.
- Ackroyd, S. (1992). *Data collection in context*. Longman Group United Kingdom.
- Agarwal, M., & Gael, S. (2014). Expert System and it ' s Requirement Engineering Process.
- Ahmad, A., & Ghazali, M. A. (2007). Documenting requirements traceability information for small projects. In *Multitopic Conference, 2007. INMIC 2007. IEEE International* (pp. 1–5). IEEE.
- Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., & Shaham-Gafni, Y. (2006). Model traceability. *IBM Systems Journal*, 45(3), 515–526.
- Alghazzawi, D. M., Siddiqui, S. T., Bokhari, M. U., & Hamatta, H. S. A. (2014). Selecting Appropriate Requirements Management Tool for Developing Secure Enterprises Software. *International Journal of Information Technology and Computer Science*, 6(4), 49–55. <http://doi.org/10.5815/ijitcs.2014.04.06>
- Ali, N. (2010). *Traceability Improvement for Software Miniaturization*.
- Ali, N., Guéhéneuc, Y.-G., & Antoniol, G. (2012). Factors impacting the inputs of

- traceability recovery approaches. In *Software and Systems Traceability* (pp. 99–127). Springer.
- Ali, N., Gueneuc, Y.-G., & Antoniol, G. (2013). Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. *Software Engineering, IEEE Transactions on*, 39(5), 725–741.
- Ali, N., Sharafi, Z., Guéhéneuc, Y.-G., & Antoniol, G. (2015). An empirical study on the importance of source code entities for requirements traceability. *Empirical Software Engineering*, 20(2), 442–478.
- Arkley, P., Mason, P., & Riddle, S. (2002). Position paper: Enabling traceability. In *Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, Edinburgh, Scotland (September 2002)* (pp. 61–65). Citeseer.
- Arkley, P., & Riddle, S. (2005). Overcoming the traceability benefit problem. In *13th IEEE International Conference on Requirements Engineering (RE'05)* (pp. 385–389). IEEE.
- Athira, B., & Samuel, P. (2011). Traceability Matrix for Regression Testing in Distributed Software Development. In *Advances in Computing and Communications* (pp. 80–87). Springer.
- Atieno, O. (2009). An analysis of the strengths and limitations of qualitative and quantitative research paradigms. *Problems of Education in the 21st Century*, 13(1), 13–38.
- Attarha, M., & Modiri, N. (2011). Focusing on the importance and the role of requirement engineering. *The 4th International Conference on Interaction Sciences*, 181–184.
- Auerbach, C., & Silverstein, L. B. (2003). *Qualitative data: An introduction to coding*

and analysis. NYU press.

Azhar, D., Mendes, E., & Riddle, P. (2012). A systematic review of web resource estimation. In *Proceedings of the 8th International Conference on Predictive Models in Software Engineering* (pp. 49–58). ACM.

Bacchelli, A., Lanza, M., & Robbes, R. (2010). Linking e-mails and source code artifacts. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1* (pp. 375–384). ACM.

Balvanes, M., & Caputi, P. (2001). Introduction to quantitative research methods. London: Sage Publications.

Bano, M., Imtiaz, S., Ikram, N., Niazi, M., & Usman, M. (2012). Causes of Requirement Change – A Systematic Literature Review, 22–31.

Bashir, M. F., & Qadir, M. A. (2006). Traceability techniques: A critical study. *10th IEEE International Multitopic Conference 2006, INMIC*, 265–268.  
<http://doi.org/10.1109/INMIC.2006.358175>

Basit, T. (2003). Manual or electronic? The role of coding in qualitative data analysis. *Educational Research*, 45(2), 143–154.

Bauer, M. W., & Gaskell, G. (2000). *Qualitative researching with text, image and sound: A practical handbook for social research*. Sage.

Blaauboer, F. A. (2006). Deciding to adopt traceability in practice: influencing this decision.

Blaauboer, F., Sikkil, K., & Aydin, M. N. (2007). Deciding to adopt requirements traceability in practice. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4495 LNCS, 294–308. Retrieved from <http://www.scopus.com/inward/record.url?eid=2-s2.0-38149045785&partnerID=tZOtx3y1>

- Bonaccorsi, A., Giannangeli, S., & Rossi, C. (2006). Entry strategies under competing standards: Hybrid business models in the open source software industry. *Management Science*, 52(7), 1085–1098.
- Borges, R. V., Garcez, A. d'Avila, Lamb, L. C., & Nuseibeh, B. (2011). Learning to adapt requirements specifications of evolving systems: (NIER track). *2011 33rd International Conference on Software Engineering (ICSE)*, 856–859.  
<http://doi.org/10.1145/1985793.1985924>
- Bouillon, E., Mäder, P., & Philippow, I. (2013). A survey on usage scenarios for requirements traceability in practice. In *Requirements Engineering: Foundation for Software Quality* (pp. 158–173). Springer.
- Burnay, C., Jureta, I., & Faulkner, S. (2015). Towards a Model of Topic Relevance during requirements elicitation-Preliminary results. In *Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on* (pp. 151–158). IEEE.
- Burns, N., & Grove, S. K. (2003). *Understanding nursing research* (3rd Ed.). Philadelphia. Saunders.
- Casey, V., & Mc Caffery, F. (2011). Med-Trace: traceability assessment method for medical device software development.
- Chakraborty, A., Baowaly, M. K., Arefin, A., & Bahar, A. N. (2012). The role of requirement engineering in software development life cycle. *Journal of Emerging Trends in Computing and Information Sciences*, 3(5), 723–729.
- Chaos, E. (2001). The Standish Group International. Inc.
- Charreire, S., & Durieux, F. (2001). Exploring and testing. *RA. Thietart et Al.(a Cura Di), Doing Management Research: A Comprehensive Guide, London: Sage.*
- Chen, X., Hosking, J., & Grundy, J. (2011). A combination approach for enhancing

- automated traceability:(NIER track). In *Software Engineering (ICSE), 2011 33rd International Conference on* (pp. 912–915). IEEE.
- Cheng, B. H. C., & Atlee, J. M. (2007). Research directions in requirements engineering. In *2007 Future of Software Engineering* (pp. 285–303). IEEE Computer Society.
- Chisaka, C., & Vakalisa, N. C. G. (2000). Gathering and analysis of data using qualitative methods in Education. *Gauteng: Rand Afrikaans University*.
- Clark, V. L. P., & Creswell, J. W. (2014). *Understanding research: A consumer's guide*. Pearson Higher Ed.
- Cleland-Huang, J. (2005). Toward improved traceability of non-functional requirements. In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering* (pp. 14–19). ACM.
- Cleland-Huang, J. (2006). Just enough requirements traceability. In *Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International* (Vol. 1, pp. 41–42). IEEE.
- Cleland-Huang, J., Chang, C. K., & Christensen, M. (2003). Event-based traceability for managing evolutionary change. *Software Engineering, IEEE Transactions on*, 29(9), 796–810.
- Cleland-Huang, J., Chang, C. K., Sethi, G., Javvaji, K., Hu, H., & Xia, J. (2002). Automating speculative queries through event-based requirements traceability. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on* (pp. 289–296). IEEE.
- Cleland-Huang, J., Czauderna, A., Dekhtyar, A., Gotel, O., Hayes, J. H., Keenan, E., ... Shin, Y. (2011). Grand challenges, benchmarks, and TraceLab: developing infrastructure for the software traceability research community. In *Proceedings of*

*the 6th international workshop on traceability in emerging forms of software engineering* (pp. 17–23). ACM.

Cleland-huang, J., Gotel, O., Hayes, J. H., Mäder, P., Zisman, A., & Keyes, M. (2014). Software Traceability : Trends and Future Directions. *Proceedings of the on Future of Software Engineering (FOSE'14)*, 55–69.

Cleland-Huang, J., Hayes, J. H., & Domel, J. M. (2009). Model-based traceability. In *Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering* (pp. 6–10). IEEE Computer Society.

Cleland-huang, J., Hayes, J. H., & Domel, J. M. (2009). Model-Based Traceability School of Computing. *Management, Copyright*, 6–10.  
<http://doi.org/10.1109/TEFSE.2009.5069575>

Cleland-Huang, J., & Schmelzer, D. (2003). Dynamically tracing non-functional requirements through design pattern invariants. In *Workshop on Traceability in Emerging Forms of Software Engineering, in conjunction with IEEE International Conference on Automated Software Engineering* (Vol. 10).

Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., & Christina, S. (2005). Goal-centric traceability for managing non-functional requirements. In *Proceedings of the 27th international conference on Software engineering* (pp. 362–371). ACM.

Coakes, S. J., & Steed, L. (2009). *SPSS: Analysis without anguish using SPSS version 14.0 for Windows*. John Wiley & Sons, Inc.

Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Cuddeback, D., Dekhtyar, A., & Hayes, J. H. (2010). Automated requirements traceability: The study of human analysts. In *Requirements Engineering*



- Conference (RE), 2010 18th IEEE International* (pp. 231–240). IEEE.
- Cysneiros, L. M. (2007). Evaluating the Effectiveness of Using Catalogues to Elicit Non-Functional Requirements. In *WER* (pp. 107–115).
- da Silva, A. R. (2014). Quality of requirements specifications: a preliminary overview of an automatic validation approach. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (pp. 1021–1022). ACM.
- De Lucia, A., Fasano, F., & Oliveto, R. (2008). Traceability management for impact analysis. In *Frontiers of Software Maintenance, 2008. FoSM 2008.* (pp. 21–30). IEEE.
- De Lucia, A., Marcus, A., Oliveto, R., & Poshyvanyk, D. (2012). Information retrieval methods for automated traceability recovery. In *Software and systems traceability* (pp. 71–98). Springer.
- Dick, a J. J. (2012). Evidence-based development - coupling structured argumentation with requirements development. *System Safety, Incorporating the Cyber Security Conference 2012, 7th IET International Conference on*, 1–5.  
<http://doi.org/10.1049/cp.2012.1498>
- Diev, S. (2007). Requirements development as a modeling activity. *ACM SIGSOFT Software Engineering Notes*, 32(2), 1–3.
- Dybä, T., Kitchenham, B., & Jorgensen, M. (2005). Evidence-based software engineering for practitioners. *Software, IEEE*, 22(1), 58–65.
- Egyed, A., & Grunbacher, P. (2002). Automating requirements traceability: Beyond the record & replay paradigm. In *Automated Software Engineering, 2002. Proceedings. ASE 2002. 17th IEEE International Conference on* (pp. 163–171). IEEE.
- Egyed, A., Grünbacher, P., Heindl, M., & Biffl, S. (2009). Value-based requirements

traceability: Lessons learned. In *Design requirements engineering: a ten-year perspective* (pp. 240–257). Springer.

Elliott Sr, R. A., & Allen, E. B. (2013). A methodology for creating an IEEE standard 830-1998 software requirements specification document. *Journal of Computing Sciences in Colleges*, 29(2), 123–131.

Fatwanto, A. (2013). Software requirements specification analysis using natural language processing technique. *2013 International Conference on QiR*, 105–110.  
<http://doi.org/10.1109/QiR.2013.6632546>

Felderer, M., & Beer, A. (2013). Using defect taxonomies for requirements validation in industrial projects. *2013 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*, 296–301.  
<http://doi.org/10.1109/RE.2013.6636733>

Ferreira, D., & da Silva, A. R. (2008). A requirements specification case study with ProjectIT-studio/requirements. In *Proceedings of the 2008 ACM symposium on Applied computing* (pp. 656–657). ACM.

Fontana, A., & Frey, J. H. (2005). The Interview: From Neutral Stance To Political Involvement. In the book of The Sage of Handbook of Qualitative Research. United Kingdom: Sage Publications, Inc.

Fricker, S. A., Grau, R., & Zwingli, A. (2015). Requirements Engineering : Best Practice Requirements Engineering State-of-Art.

Goknil, A., Kurtev, I., van den Berg, K., & Spijkerman, W. (2014). Change impact analysis for requirements: A metamodeling approach. *Information and Software Technology*, 56(8), 950–972. <http://doi.org/10.1016/j.infsof.2014.03.002>

Gotel, O. C., & Finkelstein, A. C. W. (1994). An Analysis of the Requirements Traceability Problem Imperial College of Science , Technology & Medicine

Department of Computing , 180 Queen ' s Gate, 94–101.

- Gotel, O. C. Z., & Finkelstein, A. C. W. (1994). An analysis of the requirements traceability problem. In *Requirements Engineering, 1994., Proceedings of the First International Conference on* (pp. 94–101). IEEE.
- Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P., ... Maletic, J. (2012). The grand challenge of traceability (v1. 0). In *Software and Systems Traceability* (pp. 343–409). Springer.
- Gough, D., Oliver, S., & Thomas, J. (2012). *An introduction to systematic reviews*. Sage.
- Han, K., Youn, J., & Cho, J. (2014). A Functional Requirement Traceability Management Methodology for Model-based Testing Framework of Automotive Embedded System, (c), 46–51.
- Hannay, J. E., Sjøberg, D. I. K., & Dybå, T. (2007). A systematic review of theory use in software engineering experiments. *Software Engineering, IEEE Transactions on*, 33(2), 87–107.
- Harfoushi, O., Fawwaz, B. A., Obiedat, R., Faris, H., & Al-Sayyed, R. (2012). Usability Assessment of the Government Web Services in the Hashemite Kingdom of Jordan. *Journal of American Science*, 8(12), 12.
- Haron, A. (2012). Understanding the Requirement Engineering for Organization : The Challenges, 561–567.
- Hassnain, M. (2015). A Comparative Study on Traceability Approaches in Software development Life Cycle. *ITEE Journal Information Technology & Electrical Engineering*, (2), 4.
- Hayes, J. H., Dekhtyar, A., Sundaram, S. K., Holbrook, E. A., Vadlamudi, S., & April, A. (2007). REquirements TRacing On target (RETRO): improving software

maintenance through traceability recovery. *Innovations in Systems and Software Engineering*, 3(3), 193–202.

Heindl, M., & Biffl, S. (2005). A case study on value-based requirements tracing. In *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering* (pp. 60–69). ACM.

Hill, R. (1998). What sample size is “enough” in internet survey research. *Interpersonal Computing and Technology: An Electronic Journal for the 21st Century*, 6(3-4), 1–12.

Hong, Y., Kim, M., & Lee, S.-W. (2010). Requirements management tool with evolving traceability for heterogeneous artifacts in the entire life cycle. In *Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on* (pp. 248–255). IEEE.

Huang, R., Berenbach, B., & Clark, S. (2007). Best practices for automated traceability.

Jaber, K., Sharif, B., & Liu, C. (2013). A study on the effect of traceability links in software maintenance. *Access, IEEE*, 1, 726–741.

Javed, M. A., & Zdun, U. (2014). A systematic literature review of traceability approaches between software architecture and source code. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (p. 16). ACM.

Jiang, L., Eberlein, A., Far, B. H., & Mousavi, M. (2008). A methodology for the selection of requirements engineering techniques. *Software & Systems Modeling*, 7(3), 303–328.

Juergens, E., Deissenboeck, F., Feilkas, M., Hummel, B., Schaetz, B., Wagner, S., ... Streit, J. (2010). Can clone detection support quality assessments of requirements

- specifications? In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2* (pp. 79–88). ACM.
- Kang, S., Kim, J., Kang, S., & Eom, S. (2014). A Formal Representation of Platform Feature-to-Requirement Traceability for Software Product Line Development. *2014 IEEE 38th Annual Computer Software and Applications Conference*, 211–218. <http://doi.org/10.1109/COMPSAC.2014.29>
- Katta, V., & Stålhane, T. (2014). Survey Protocol : Traceability during Development of Systems with Safety and Security Implications - Importance , Tools , and Challenges.
- Khan, K., Kumar, P. V. V., Ahmad, A., Riaz, T., Anwer, W., Suleman, M., ... Chaitanya, a. V. K. (2011). Requirement Development Life Cycle: The Industry Practices. *2011 Ninth International Conference on Software Engineering Research, Management and Applications*, 12–16. <http://doi.org/10.1109/SERA.2011.38>
- Khan, K., Kunz, R., Kleijnen, J., & Antes, G. (2011). *Systematic reviews to support evidence-based medicine*. CRC Press.
- Khan, M. N. A., Khalid, M., & ul Haq, S. (2013). Review of requirements management issues in software development. *International Journal of Modern Education and Computer Science (IJMECS)*, 5(1), 21.
- Kirova, V., Kirby, N., Kothari, D., & Childress, G. (2008). Effective requirements traceability: Models, tools, and practices. *Bell Labs Technical Journal*, 12(4), 143–157.
- Kitchenham, B. A., Dyba, T., & Jorgensen, M. (2004). Evidence-based software engineering. In *Proceedings of the 26th international conference on software engineering* (pp. 273–281). IEEE Computer Society.

- Kitchenham, B., & Charters, S. (n.d.). Guidelines for performing systematic literature reviews in software engineering. 2007. URL [Http://www. Dur. Ac. uk/ebse/resources/Systematic-Reviews-5-8. Pdf](http://www.Dur.Ac.uk/ebse/resources/Systematic-Reviews-5-8.Pdf).
- Klimpke, L., & Hildenbrand, T. (2009). Towards end-to-end traceability: Insights and implications from five case studies. In *Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on* (pp. 465–470). IEEE.
- Kong, L., & Yuan, T. (2009). Extension Features-Driven Use Case Model for requirement traceability. In *Computer Science & Education, 2009. ICCSE '09. 4th International Conference on* (pp. 866–870). IEEE.
- Kothari, C. R. (2009). *Research methodology: Methods and techniques New Delhi: New Age International Publishers*. ISBN 978-81-224-15222-3.
- Kumar, R. (2011). *Research methodology: A step-by-step guide for Beginners* (3rd ed.).
- Lago, P., Muccini, H., & van Vliet, H. (2009). A scoped approach to traceability management. *Journal of Systems and Software*, 82(1), 168–182. <http://doi.org/10.1016/j.jss.2008.08.026>
- Laplante, P. A. (2013). *Requirements engineering for software and systems*. CRC Press.
- Lin, J., Lin, C. C., Cleland-Huang, J., Settini, R., Amaya, J., Bedford, G., ... Zou, X. (2006). Poirot: A distributed tool supporting enterprise-wide automated traceability. In *Requirements Engineering, 14th IEEE International Conference* (pp. 363–364). IEEE.
- Mader, P., & Egyed, A. (2012). Assessing the effect of requirements traceability for software maintenance. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on* (pp. 171–180). IEEE.
- Mäder, P., & Egyed, A. (2014). Do developers benefit from requirements traceability when evolving and maintaining a software system? *Empirical Software*

*Engineering*, 1–29.

Mäder, P., & Gotel, O. (2012). Towards automated traceability maintenance. *Journal of Systems and Software*, 85(10), 2205–2227.

Mäder, P., Gotel, O., Kuschke, T., & Philippow, I. (2008). traceMaintainer-Automated Traceability Maintenance. In *International Requirements Engineering, 2008. RE'08. 16th IEEE* (pp. 329–330). IEEE.

Mader, P., Gotel, O., & Philippow, I. (2008). Rule-based maintenance of post-requirements traceability relations. In *International Requirements Engineering, 2008. RE'08. 16th IEEE* (pp. 23–32). IEEE.

Mäder, P., Gotel, O., & Philippow, I. (2009a). Motivation matters in the traceability trenches. In *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International* (pp. 143–148). IEEE.

Mäder, P., Gotel, O., & Philippow, I. (2009b). Semi-automated traceability maintenance: An architectural overview of traceMAINTAINER. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on* (pp. 425–426). IEEE.

Mäder, P., Gotel, O., & Philippow, I. (2009c). traceMAINTAINER: A Tool for the Semi-automated Maintenance of Model Traceability. *Submitted*), Enschede, Netherlands (June 2009).

Mader, P., Jones, P. L., Zhang, Y., & Cleland-Huang, J. (2013). Strategic traceability for safety-critical projects. *Software, IEEE*, 30(3), 58–66.

Mahmoud, A. (2015). An information theoretic approach for extracting and tracing non-functional requirements. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International* (pp. 36–45). IEEE.

Maletic, J. I., Munson, E. V, Marcus, A., & Nguyen, T. N. (2003). Using a hypertext

- model for traceability link conformance analysis. In *Proc. of the Int. Workshop on Traceability in Emerging Forms of Software Engineering* (pp. 47–54).
- Malhotra, N. K. (2008). *Marketing research: An applied orientation*, 5/e. Pearson Education India.
- Manapian, A., & Prompoon, N. (2014). Software time estimation model for requirements change based on software prototype profiles using an analogy estimation method. In *Computer Science and Engineering Conference (ICSEC), 2014 International* (pp. 366–371). IEEE.
- Marnewick, A. (2014). The effect of requirements engineering on the success of system implementation: a comparative case study.
- Marshall, C., & Rossman, G. B. (1999). *Designing qualitative research*. Sage publications.
- Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Sage.
- Mirakhorli, M., & Cleland-Huang, J. (2012). Tracing non-functional requirements. In *Software and Systems Traceability* (pp. 299–320). Springer.
- Mohebzada, J. G., Ruhe, G., & Eberlein, A. (2012). Systematic Mapping of Recommendation Systems for Requirements Engineering, 200–209.
- Morckos, M. (2011). Requirements Traceability. *Report for School of Computer Science, University of Waterloo, Waterloo*.
- Muijs, D. (2010). *Doing quantitative research in education with SPSS*. Sage.
- Murta, L. G. P., Van Der Hoek, A., & Werner, C. M. L. (2006). ArchTrace: policy-based support for managing evolving architecture-to-implementation traceability links. In *Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM*



*International Conference on* (pp. 135–144). IEEE.

Murta, L. G. P., van der Hoek, A., & Werner, C. M. L. (2008). Continuous and automated evolution of architecture-to-implementation traceability links. *Automated Software Engineering*, 15(1), 75–107.

Nair, S., De La Vara, J. L., & Sen, S. (2013). A review of traceability research at the requirements engineering conference@21. *2013 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*, 222–229. <http://doi.org/10.1109/RE.2013.6636722>

Naz, H., Motla, Y. H., Asghar, S., Abbas, M. A., & Khatoon, A. (2013). Effective usage of AI technique for requirement change management practices. *2013 5th International Conference on Computer Science and Information Technology, CSIT 2013 - Proceedings*, 121–125. <http://doi.org/10.1109/CSIT.2013.6588769>

Niessink, F., & Van Vliet, H. (1998). Two case studies in measuring software maintenance effort. In *Software Maintenance, 1998. Proceedings., International Conference on* (pp. 76–85). IEEE.

Noor, N. M. (2011). *Writing Research and Thesis Proposals: Guidelines and Examples*. Pusat Penerbitan Universiti Teknologi Mara.

O'Connor, A. M., Anderson, K. M., Goodell, C. K., & Sargeant, J. M. (2014). Conducting systematic reviews of intervention questions I: Writing the review protocol, formulating the question and searching the literature. *Zoonoses and Public Health*, 61(S1), 28–38.

Ohashi, K., Kurihara, H., Tananaka, Y., & Yamamoto, R. (2011). A means of establishing traceability based on a UML model in business application development. In *Requirements Engineering Conference (RE), 2011 19th IEEE International* (pp. 279–284). IEEE.

- Omoronyia, I., Sindre, G., Biffi, S., & Stålhane, T. (2011). Understanding Architectural Elements from Requirements Traceability Networks. In *Relating Software Requirements and Architectures* (pp. 61–83). Springer.
- Ooi, S. M., Lim, R., & Lim, C. C. (2014). An integrated system for end-to-end traceability and requirements test coverage. In *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on* (pp. 45–48). IEEE.
- Ozkaya, I., & Akin, Ö. (2007). Tool support for computer-aided requirement traceability in architectural design: The case of DesignTrack. *Automation in Construction*, 16(5), 674–684.
- Pandanaboyana, S., Sridharan, S., Yannelli, J., & Hayes, J. H. (2013). REquirements TRacing On target (RETRO) enhanced with an automated thesaurus builder: An empirical study. In *Traceability in Emerging Forms of Software Engineering (TEFSE), 2013 International Workshop on* (pp. 61–67). IEEE.
- Pandey, D., & Suman, U. (2012). The Usefulness of RE Practices in S/W Development. *Journal of Global Research in Computer Science*, 3(9), 56–69.
- Pandey, D., Suman, U., & Ramani, A. K. (2010). An effective requirement engineering process model for software development and requirements management. In *Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference on* (pp. 287–291). IEEE.
- PC, A., & Prabhu, B. (2012). Integrating Requirements Engineering and User Experience Design in Product Life Cycle Management, 12–17.
- Petticrew, M., & Roberts, H. (2008). *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons.
- Pinheiro, F. A. C. (2004). Requirements traceability. In *Perspectives on software requirements* (pp. 91–113). Springer.

- Raja, U. A. (2009). Empirical studies of requirements validation techniques. *2009 2nd International Conference on Computer, Control and Communication, IC4 2009*.  
<http://doi.org/10.1109/IC4.2009.4909209>
- Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. *Software Engineering, IEEE Transactions on*, 27(1), 58–93.
- Ramesh, B., Stubbs, C., Powers, T., & Edwards, M. (1997). Requirements traceability : Theory and practice, 3, 397–415.
- Regan, G., McCaffery, F., McDaid, K., & Flood, D. (2012). The barriers to traceability and their potential solutions: towards a reference framework. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on* (pp. 319–322). IEEE.
- Regan, G., McCaffery, F., McDaid, K., & Flood, D. (2014). The Development and Validation of a Traceability Assessment Model. In *Software Process Improvement and Capability Determination* (pp. 72–83). Springer.
- Rempel, P., Mader, P., & Kuschke, T. (2013). An empirical study on project-specific traceability strategies. *2013 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*, 195–204.  
<http://doi.org/10.1109/RE.2013.6636719>
- Richards, L., & Morse, J. M. (2012). *Readme first for a user's guide to qualitative methods*. Sage.
- Rikhari, P., & Kumar, A. (2012). Validating the Change Management Process for Managing Changing Requirements with the help of a Real Life Project. *International Journal of Computer Applications*, 58(7).
- Rochimah, S., Wan Kadir, W. M. N., & Abdullah, A. H. (2007). An Evaluation of Traceability Approaches to Support Software Evolution. *International Conference*

*on Software Engineering Advances (ICSEA 2007)*, (October), 19–19.

<http://doi.org/10.1109/ICSEA.2007.17>

- Rosmadi, N. A., Ahmad, S., & Abdullah, N. (2015). The Relevance of Software Requirement Defect Management to Improve Requirements and Product Quality: A Systematic Literature Review. In *Pattern Analysis, Intelligent Security and the Internet of Things* (pp. 95–106). Springer.
- Sadiq, M., & Jain, S. K. (2014). Stakeholder identification method in goal oriented requirements elicitation process. In *Requirements Prioritization and Communication (RePriCo), 2014 IEEE 5th International Workshop on* (pp. 25–33). IEEE.
- Saiedian, H., Kannenberg, A., & Morozov, S. (2013). A streamlined, cost-effective database approach to manage requirements traceability. *Software Quality Journal*, 21(1), 23–38.
- Saldaña, J. (2015). *The coding manual for qualitative researchers*. Sage.
- Salvador, C., Nakasone, A., & Pow-Sang, J. A. (2014). A systematic review of usability techniques in agile methodologies. In *Proceedings of the 7th Euro American Conference on Telematics and Information Systems* (p. 17). ACM.
- Sarkan, H. M., Ahmad, T. P. S., & Bakar, A. A. (2011). Using JIRA and Redmine in requirement development for agile methodology. In *Software Engineering (MySEC), 2011 5th Malaysian Conference in* (pp. 408–413). IEEE.
- Schwarz, H., Ebert, J., & Winter, A. (2010a). Graph-based traceability: A comprehensive approach. *Software and Systems Modeling*, 9(4), 473–492.  
<http://doi.org/10.1007/s10270-009-0141-4>
- Schwarz, H., Ebert, J., & Winter, A. (2010b). Graph-based traceability: a comprehensive approach. *Software & Systems Modeling*, 9(4), 473–492.

- Sekaran, U., & Bougie, R. (2010). Research methods for business: A skill building approach. Wiley. London.
- Shahid, M., Ibrahim, S., & Mahrin, M. N. (2011). An evaluation of requirements management and traceability tools. *World Academy of Science, Engineering and Technology*, 78(6), 596–601. Retrieved from <http://www.scopus.com/inward/record.url?eid=2-s2.0-84855217455&partnerID=40&md5=63ec72af99df5960c8e026199d2ddf0f>
- Shahid, M., Ibrahim, S., & Mahrin, M. N. (2011). An Evaluation of Requirements Management and Traceability Tools. *World Academy of Science, Engineering and Technology, WASET*.
- Sharma, R., Bhatia, J., & Biswas, K. K. (2014). Machine learning for constituency test of coordinating conjunctions in requirements specifications. *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering - RAISE 2014*, 25–31. <http://doi.org/10.1145/2593801.2593806>
- Sharma, S., & Pandey, S. K. (2014). Requirements elicitation: Issues and challenges. In *Computing for Sustainable Global Development (INDIACom), 2014 International Conference on* (pp. 151–155). IEEE.
- Sherba, S. A., Anderson, K. M., & Faisal, M. (2003). A framework for mapping traceability relationships. In *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering* (pp. 32–39).
- Shubhamangala, B. R., Rao, L. M., Dakshinamurthy, A., & Singh, C. G. L. (2012). Ability based domain specific training: a pragmatic solution to poor requirement engineering in CMM level 5 companies. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on* (Vol. 3, pp. 459–464). IEEE.

Silva Souza, V. E., Lapouchnian, A., Robinson, W. N., & Mylopoulos, J. (2011).

Awareness requirements for adaptive systems. In *Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems* (pp. 60–69). ACM.

Smith, A. M. (2012). *Research Methodology: A Step-by-step Guide for Beginners*.

Churchill Livingstone.

Solemon, B., Sahibuddin, S., & Ghani, A. A. A. (2010). Adoption of requirements

engineering practices in Malaysian software development companies. In *Advances in Software Engineering* (pp. 141–150). Springer.

Soonsongtanee, S., & Limpiyakorn, Y. (2010). Enhancement of requirements

traceability with state diagrams. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on* (Vol. 2, pp. V2–248). IEEE.

Spanoudakis, G., & Zisman, A. (2005). Software traceability: a roadmap. *Handbook of Software Engineering and Knowledge Engineering*, 3, 395–428.

Spanoudakis, G., Zisman, A., Pérez-Minana, E., & Krause, P. (2004). Rule-based

generation of requirements traceability relations. *Journal of Systems and Software*, 72(2), 105–127.

Speziale, H., & Carpenter, D. R. (2003). *Qualitative research in nursing: Advancing the humanistic imperative*. New York. Harper and Row.

Stallinger, F., Neumann, R., Schossleitner, R., & Zeilinger, R. (2011). Linking software

life cycle activities with product strategy and economics: Extending ISO/IEC 12207 with product management best practices. In *Software Process Improvement and Capability Determination* (pp. 157–168). Springer.

Sutcliffe, A., & Sawyer, P. (2013). Requirements elicitation: Towards the unknown

unknowns. *2013 21st IEEE International Requirements Engineering Conference*,

- Thomas, S. W., Adams, B., Hassan, A. E., & Blostein, D. (2014). Studying software evolution using topic models. *Science of Computer Programming*, 80, 457–479.
- Torkar, R., Gorschek, T., Feldt, R., Svahnberg, M., Raja, U. A., & Kamran, K. (2012). Requirements traceability: a systematic review and industry case study. *International Journal of Software Engineering and Knowledge Engineering*, 22(03), 385–433.
- Vagias, W. M. (2006). Likert-type scale response anchors. *Clemson International Institute for Tourism & Research Development, Department of Parks, Recreation and Tourism Management. Clemson University*.
- Wen, B., Luo, Z., & Liang, P. (2012). Distributed and Collaborative Requirements Elicitation based on Social Intelligence. In *Web Information Systems and Applications Conference (WISA), 2012 Ninth* (pp. 127–130). IEEE.
- Wiegiers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education.
- Wiegiers, K. E. (2000). When Telepathy Won't Do: Requirements Engineering Key Practices. *Cutter IT Journal*, 13(5), 9–15.
- Willig, C., & Stainton-Rogers, W. (2007). *The SAGE handbook of qualitative research in psychology*. Sage.
- Winkler, S., & Pilgrim, J. (2010). A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling (SoSyM)*, 9(4), 529–565.
- Winkler, S., & von Pilgrim, J. (2010). A survey of traceability in requirements engineering and model-driven development. *Software & Systems Modeling*, 9(4), 529–565. <http://doi.org/10.1007/s10270-009-0145-0>

- Yousuf, F., Zaman, Z., & Ikram, N. (2008). Requirements validation techniques in GSD: A survey. *IEEE INMIC 2008: 12th IEEE International Multitopic Conference - Conference Proceedings*, 553–557.  
<http://doi.org/10.1109/INMIC.2008.4777800>
- Yozgyur, K. (2014). A proposal for a requirements elicitation tool to increase stakeholder involvement. In *Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on* (pp. 145–148). IEEE.
- Zaib, A., Chauhan, M., & Sirshar, M. (2015). A Review Analysis on Non-Functional Requirements and Causes of Failure of Projects.
- Zainol, A., & Mansoor, S. (2011). An investigation of a requirements management tool elements. *2011 IEEE Conference on Open Systems, ICOS 2011*, 59–64.  
<http://doi.org/10.1109/ICOS.2011.6079304>
- Zhang, Z. (2007). Effective requirements development-A comparison of requirements elicitation techniques. *Software Quality Management XV: Software Quality in the Knowledge Society*, E. Berki, J. Nummenmaa, I. Sunley, M. Ross and G. Staples (Ed.) British Computer Society, 225–240.
- Zhou, J. (2014a). Requirements Development and Management of Embedded Real-Time Systems, 479–484.
- Zhou, J. (2014b). Requirements development and management of embedded real-time systems. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International* (pp. 479–484). IEEE.
- Zikmund, W. G. (2003). Sample designs and sampling procedures. *Business Research Methods*, 7, 368–400.
- Zisman, A. (2012). Using rules for traceability creation. In *Software and Systems Traceability* (pp. 147–170). Springer.



Zou, X., Settimi, R., & Cleland-Huang, J. (2010). Improving automated requirements trace retrieval: a study of term-based enhancement methods. *Empirical Software Engineering*, 15(2), 119–146.



## **APPENDIX A: Questionnaire**



**College of  
Arts and Sciences (CAS)**  

---

**University Utara Malaysia (UUM)**

# **AN INVESTIGATION OF REQUIREMENTS TRACEABILITY PRACTICES IN SOFTWARE COMPANIES**

## **Preface**

Requirements Traceability (RT) is the ability to follow requirements and detect changes during the software development life cycle. These changes can be made by updating and maintaining of requirements traceability links. RT is a critical element of software development process.

This document contains questionnaire to seek requirements traceability practices in your company. The questionnaire is divided into three sections; the first section asks about participant's background, the second section asks about requirements traceability tools and the third section is interested in requirements traceability techniques. Please answer all the questions based on your knowledge, skills, and your position in your company. The questionnaire aims to investigate the requirements traceability practice in your company.

To help us better interpret your answers to the questions about software process in your company, this document begins with questions about your background in the software company. Please carefully read and answer all of the questions.

**Thank you very much for your cooperation**

Name: Jasim Mohammed Dahr

Programme: Master of Science (Information Technology)

College: College of Arts and Sciences

Email: J\_M\_D86@yahoo.com

**Note:** This questionnaire is directed to software companies for collecting data on Requirements Traceability Practices as a part of my master's dissertation.

Please record an answer to each question with one of five possible responses:

1 = **Never**, 2 = **Rarely**, 3 = **Sometimes**, 4 = **Often**, 5 = **Always**.

1. Record **Never** when:

The practice is not used or uncertain about how to answer the question.

2. Record **Rarely** when:

The practice rarely used.

3. Record **Sometimes** when:

The practice may be performed sometimes or omit under difficult circumstances.

4. Record **Often** when:

The practice often established but not always.

5. Record **Always** when:

The practice is well established and consistently performed.

## SECTION A: PARTICIPANT'S BACKGROUND

The section aims to obtain your demographic information. Please answer all questions by placing a check (✓) in the appropriate box:

Q1. Software Company Name : \_\_\_\_\_.

Q2. Your position : \_\_\_\_\_.

Q3. Email (optional): \_\_\_\_\_.

Q4. Date : \_\_\_\_\_.

Q5. Age:

<input type="checkbox"/>	18-24
<input type="checkbox"/>	25-34
<input type="checkbox"/>	35-44
<input type="checkbox"/>	44-50
<input type="checkbox"/>	50 +

Q6. Gender:

- ☐ Male
- ☐ Female

Q7. Education Level:

- ☐ Diploma
- ☐ Bachelor degree
- ☐ Master degree
- ☐ PhD
- ☐ Other (please specify): \_\_\_\_\_

Q8. Organization Level (team role):

- ☐ Project Manager
- ☐ Requirement analyst
- ☐ Designer
- ☐ Developer
- ☐ Tester
- ☐ Other (please specify): \_\_\_\_\_

Q9. Sector of the software company:

- ☐ Communication
- ☐ Financial
- ☐ Education
- ☐ Networking
- ☐ Commercial
- ☐ Other (please specify): \_\_\_\_\_

Q10. Years of experience in software development projects:

- ☐ Less than 5 years
- ☐ 5-10 years
- ☐ More than 10 years

Q11. Number of employees in the company:

- ☐ 5-10 people
- ☐ 10-50 people
- ☐ More than 50 people

Q12. Number of software projects involved in:

- ☐ Less than 10 projects
- ☐ 10-20 projects
- ☐ More than 20 projects

## SECTION B: REQUIREMENTS TRACEABILITY TOOLS

The following questions aim to obtain answers about tools of requirements traceability that are used in practice. Please answer by placing a check (✓) in the suitable choices below:

		Never ← → Always				
No.	Question	1	2	3	4	5
Q13	Requirements traceability matrix is tools used to trace requirements manually. Do you use these tools to trace the requirements?					
Q14	In maintenance of traceability relations. Do you think the use of traceMAINTAINER tool helps to tracing requirements?					
Q15	Do you think the use of DesignTrack offers traceability between architectural design and requirements?					
Q16	Do you think the use of RETRO tool helps facilitating the automated generation of RT matrices?					
Q17	Do you think DevComplete provides complete traceability among project tasks and requirements tracking?					

Comment:

---



---



---



---

## SECTION C: REQUIREMENTS TRACEABILITY TECHNIQUES

The following questions aim to obtain answers about techniques of requirements traceability that are used in practice. Please answer by placing a check (✓) in the suitable choices below:

		Never ← → Always				
No.	Question	1	2	3	4	5
Q18	Event-Based (EB) is a technique used to maintain links between requirements. Do you use EB to maintain traceability relationships?					
Q19	Information Retrieval (IR) is technique use to focus on automating the generation of traceability link. Do you use IR for this purpose?					
Q20	Rule-Based (RB) is a technique used as a method to support the automatic generation of traceability relations between documents. Do you use RB for this purpose?					

Q21	ArchTrace is an approach used to support the evaluation of traceability relation between architecture and implementation. Do you use ArchTrace for continuously updating traceability relations?					
Q22	Value-Based (VB) is a technique that provides technical support to perform requirements tracing. Do you use VB for assessing the value that traceability can provide to company?					
Q23	Scenario-Based (SB) is a technique used to create traceability links automatically. Do you use SB for this purpose?					
Q24	Hypertext-based (HTB) is a useful technique for generating traceability relationships by using open hypermedia and information integration. Do you use HB to create traceability relationships?					
Q25	Goal-Centric (GC) is a technique used to improving the traceability of non-functional requirements. Do you use GC for this purpose?					
Q26	Model-Based (MB) is a technique used to overcome on traceability problems that occur when use matrices. Do you use MB to avoid these problems?					

Comment:

---



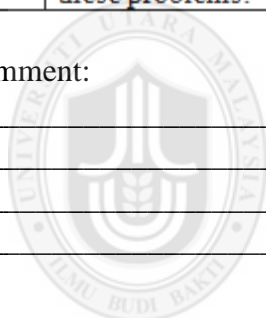
---



---



---



UUM

Universiti Utara Malaysia

## **APPENDIX B: Expert Reviewer**

**Dr. Mazni Omar** is a senior lecturer at the School of Computing, College of Arts and Sciences, Universiti Utara Malaysia. She received the BSc. degree (with honors) in information technology from Universiti Utara Malaysia, in 2000, the MSc. degree in software engineering from Universiti Teknologi Malaysia, in 2002, and the Ph.D. degree in information technology and quantitative sciences from Universiti Teknologi MARA, Malaysia, in 2012. Her current research interests include agile software development, empirical software engineering, software quality and data mining. She has been actively teaching and conducting research for the last 15 years in the area of software engineering.

**Dr. Nor Azliana Akmal Jamaludin** is a Senior Lecturer at Universiti Selangor. Her Doctoral of Philosophy (PhD) in Computer Science, specialize in Software Engineering at UTM. She is a member of Malaysian Software Engineering Interest Group, Malaysia. Her field of expertise is in software requirement, requirement engineering, analysis, system integration, e-learning, software maintenance and Software Engineering Education. Her current research interest is on techniques that can enhance skill among Software Engineering undergraduate of higher institutions using eLearning. She has been very active in scholarly journals writing and publishing citation index/impact factor journal papers.

**Dr. MAZIDA AHMAD** is a Senior Lecturer at the School of Computing, Universiti Utara Malaysia (UUM). She received the BMIS degree from International Islamic University of Malaysia, in 2001, the MSc. degree in Software Engineering from Universiti Teknologi Malaysia, in 2003, and the Ph.D. degree in Knowledge Management from Universiti Sains Malaysia, in 2010. She has taught a variety of courses in management information system, knowledge management, system analysis and design and information system development at undergraduate and postgraduate levels. Her current research interests include knowledge management, information system development and software engineering education.

### APPENDIX C: List of Software Companies

No.	Software Company Name	No.	Software Company Name
1	Pentech Solution Sdn Bhd	17	eNCoral Digital Solutions
2	Exact	18	SYN System Solutions
3	ES&S MSC Sdn Bhd	19	Maple software creation
4	st jude medical	20	Encelabs
5	Siemens Industry Software Sdn. Bhd	21	GracusSoft Solutions
6	brady co.	22	eSource Technology
7	MIMOS	23	Bogus company
8	Osram	24	Qubit Enterprise
9	e-Business Sdn. Bhd.	25	Inari Technology
10	Intel Microelectronics Sdn. Bhd.	26	Operion Ecommerce & Software Sdn Bhd
11	PriceWaterhouseCoopers Malaysia	27	FLEX software consulting Sdn Bhd
12	Eframe Solutions Sdn Bhd	28	ABS Software International
13	CSG international	29	Software Depot Sdn Bhd
14	Terato Tech Sdn Bhd	30	PS TECH
15	Profitera Corporation Sdn. Bhd	31	My Software Solutions
16	Heitech Padu		



## **APPENDIX D: Interview Questions**

**Q1:** Based on your experience, how your company applied/ exploited traceability practices?

**Q2:** Why the company did not use the requirements traceability practices?

**Q3:** Do you think that RT practices can help company to produce high quality software?

**Q4:** What kind of types of requirements that you prefer to apply the RT practices (functional, non-functional) and why?

**Q5:** At any stage of software development is required to apply traceability more than the others?

**Q6:** What are your recommendation for the companies that still not use these practices, and how can implement?



## APPENDIX E: Studies Selected of Systematic Literature Review

No.	Study title	Author(s)	Year	Type of publication	Extraction Data	Database
1	IR-based Traceability Recovery Processes: an Empirical Comparison of “One-Shot” and Incremental Processes	Lucia, Oliveto and Tortora	2008	ASE Automated Software Engineering	IR	ACM
2	How Do We Trace Requirements: An Initial Study of Analyst Behavior in Trace Validation Tasks	Kong, Hayes, Dekhtyar and Holden	2011	ICSE International Conference on Software Engineering	RETRO	ACM
3	Extraction and Visualization of Traceability Relationships between Documents and Source Code	Chen	2010	ASE Automated Software Engineering	IR	ACM
4	Traceability and Completeness Checking for AgentOriented Systems	Cysneiros and Zisman	2008	SAC Symposium on Applied Computing	RB	ACM
5	Model-Based Traceability	Cleland-Huang, Hayes and Domel	2009	IEEE Computer Society Washington, DC, USA	MB	ACM
6	ADAMS Re-Trace: Traceability Link Recovery via Latent Semantic Indexing	Lucia, Oliveto and Tortora	2008	ICSE International Conference on Software Engineering	IR	ACM
7	Linking E-Mails and Source Code Artifacts	Bacchelli, Lanza and Robbes	2010	ICSE International Conference on Software Engineering	IR	ACM

No.	Study title	Author(s)	Year	Type of publication	Extraction Data	Database
8	A survey of traceability in requirements engineering and model-driven development	Winkler and Pilgrim	2010	Chapter book-Software and Systems Modeling (SoSyM)	RTM	ACM
9	Traceability Improvement For Software Miniaturization	Ali	2010	Chapter book	IR, EB, HTB, RB, SB and VB	Google scholar
10	An Evaluation of Requirements Management and Traceability Tools	Shahid, Ibrahim and Mahrin	2011	World Academy of Science, Engineering and Technology, WASET	Retro, DevComplete and DesignTrack	Google scholar
11	traceMAINTAINER: A Tool for the Semi-automated Maintenance of Model Traceability	Mader, Gotel and Philippow	2009	Enschede, Netherlands (June 2009). 2009.	traceMAINTAINER	Google scholar
12	Rule-Based Maintenance of Post-Requirements Traceability Relations	Mader, Gotel and Philippow	2008	International requirements Engineering, 2008. RE '08. 16th IEEE	RB and traceMAINTAINER	IEEE
13	Requirements Management Tool with Evolving Traceability for Heterogeneous Artifacts in the Entire Life Cycle	Hong, Kim and Lee	2010	Software Engineering Research, Management and Applications (SERA)	ArchTrace	IEEE
14	Tracing Requirements In A Model-Based Testing Approach	Abbors, Trusçan and Lilius	2009	Advances in System Testing and Validation Lifecycle, 2009.	MB	IEEE
15	A means of establishing traceability based on a UML model in business application development	Ohashi, Kurihara, Tananaka and Yamamoto	2011	Requirements Engineering Conference (RE), 2011 19th IEEE International	MB	IEEE

No.	Study title	Author(s)	Year	Type of publication	Extraction Data	Database
16	A Combination Approach for Enhancing Automated Traceability (NIER Track)	Chen, Hosking and Grundy	2011	Software Engineering (ICSE), 2011 33rd International Conference	IR	IEEE
17	Goal-Centric Traceability: Using Virtual Plumblines to Maintain Critical Systemic Qualities	Cleland-Huang, Marrero and Berenbach	2008	Software Engineering, IEEE Transactions	GC	IEEE
18	REquirements TRacing On target (RETRO) Enhanced with an Automated Thesaurus Builder: An Empirical Study	Pandanaboyana, Sridharan, Yannelli and Hayes	2013	Traceability in Emerging Forms of Software Engineering (TEFSE), 2013 International Workshop	RETRO and IR	IEEE
19	Automated Requirements Traceability: the Study of Human Analysts	Cuddeback, Dekhtyar and Hayes	2010	Requirements Engineering Conference (RE), 2010 18th IEEE International	RTM, Retro and IR	IEEE
20	traceMaintainer – Automated Traceability Maintenance	Mäder, Gotel, Kuschke and Philippow	2008	16th IEEE International Requirements Engineering Conference	traceMAINTAINER	IEEE
21	Semi-automated Traceability Maintenance: An Architectural Overview of traceMaintainer	Mäder, Gotel and Philippow	2009	Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on (pp. 425-426).	traceMAINTAINER	IEEE
22	Towards automated traceability maintenance	Mäder and Gotel	2012	The Journal of Systems and Software	EB, RB and traceMAINTAINER	ScienceDirect

No.	Study title	Author(s)	Year	Type of publication	Extraction Data	Database
23	Information Retrieval Methods for Automated Traceability Recovery	Lucia, Marcus, Oliveto and Poshyvanyk	2012	Chapter book- Software and Systems Traceability	IR	Springer
24	Continuous and automated evolution of architecture-to-implementation traceability links	Murta, van der Hoek and Werner	2008	Journal- Automated Software Engineering	ArchTrace	Springer
25	Value-Based Requirements Traceability: Lessons Learned	Egyed, Grünbacher, Heindl and Biffel	2009	Chapter book- Design Requirements Engineering: A Ten-Year Perspective	VB	Springer
26	Understanding Architectural Elements from Requirements Traceability Networks	Omoronyia, Sindre, Biffel and Stalhane	2011	Chapter book- Relating Software Requirements and Architectures	EB	Springer
27	XTraQue: traceability for product line systems	Jirapanthong and Zisman	2009	Software & Systems Modeling 8, no. 1 (2009): 117-144.	RB	Springer
28	Assessing IR-based traceability recovery tools through controlled experiments	Lucia, Oliveto and Tortora	2009	Empirical Software Engineering 14, no. 1 (2009): 57-92.	IR	Springer
29	Supporting requirements to code traceability through refactoring	Mahmoud and Niu	2014	Requirements Engineering. 2014 Sep 1;19(3):309-29.	IR	Springer
30	An empirical study on the importance of source code entities for requirements traceability	Ali, Sharafi, Guéhéneuc and Antoniol	2015	Journal-Empirical Software Engineering	IR	Springer

No.	Study title	Author(s)	Year	Type of publication	Extraction Data	Database
31	Factors Impacting the Inputs of Traceability Recovery Approaches	Ali, Guéhéneuc, and Antoniol	2012	Chapter book- Software and Systems Traceability	SB, RB, EB, HTB and IR	Springer
32	Improving automated requirements trace retrieval: a study of term-based enhancement methods	Zou, Settimi and Cleland-Huang	2010	Empirical Software Engineering, 15(2), pp.119-146.	IR	Springer
33	Using Rules for Traceability Creation	Zisman	2012	Chapter book, In Software and Systems Traceability, pp. 147-170. Springer London, 2012.	RB	Springer
34	Tracing Non-Functional Requirements	Mirakhorli and Cleland-Huang	2012	Chapter book- Software and Systems Traceability	GC	Springer
35	Traceability Matrix for Regression Testing in Distributed Software Development	Athira and Samuel	2011	Chapter book- Advances in Computing and Communications	RTM	Springer
36	Recovering Traceability Links between Business Activities and Software Components	Aversano, Marulli and Tortorella	2010	Chapter book- ENTERprise Information Systems	IR	Springer
37	Enabling automated traceability maintenance through the upkeep of traceability relations	Mader, Gotel and Philippow	2009	Chapter book- Model Driven Architecture - Foundations and Applications	traceMAINTAINER	Springer