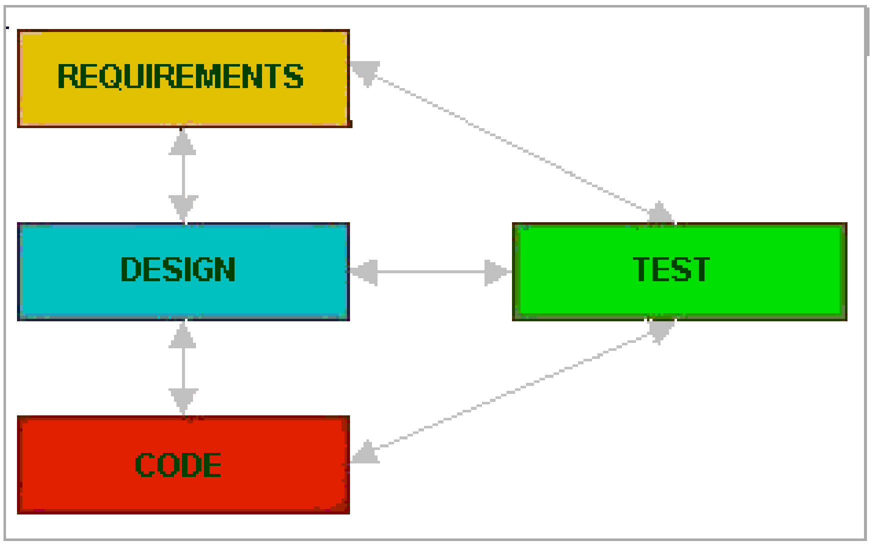
UNIVERSITY OF GOTHENBURG



Traceability in Agile software projects

Master of Science Thesis in Software Engineering & Management

VUONG HOANG DUC

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, May 2013

Traceability in Agile software projects

VUONG.HOANG DUC

VUONG. HOANG DUC, May 2013.

Examiner: CHRISTIAN.BERGER

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

ABSTRACT

Context: Software applications have been penetrating every corner of our daily life in the past decades. This condition demands high quality software. Traceability activities have been recognized as important factors supporting various activities during the development process of a software system with the aim of improving software quality. Over the past decades, software traceability has been discussed in literature and there are many approaches proposed to achieve traceability. However, these current literature mainly discussed traceability in traditional software development process and very few studies are discussing traceability issues in Agile software process.
Objective: This study aims to investigate the important types of traceability, benefits against challenges when conducting traceability in an Agile software project. This paper also aims to identify what mechanisms are used and how they work and help companies to achieve traceability in Agile software projects.
Method: This study is conducted by interviewing 12 Agile software companies and reviewing 20 primary papers related to this research topic.
Result: Rationale, Contribute and Refinement are three most important types of traceability. Traditional traceability tools such as: excel, product backlog, communication, etc are used together with automated software tools. Automated tools could be commercial tools made or self-developed by companies. Difficulties related to knowledge, experience, cooperation or commitment within the development team seem to be the most challenging. Doing traceability could help the software company gain the customer focus and guarantee or safety for its software system.
Keywords: traceability goals/purposes, agile/lean traceability, type of traceability, Agile principles.

ACKNOWLEDGEMENT

This master thesis would not have been possible without the sincere help and contribution of several people. I would like to take this opportunity to express my gratitude to them.

First of all, I would like to express my sincere gratitude to my supervisor Professor Tony Gorschek for providing me with valuable guidance and advice throughout this thesis. Not only did Professor Tony guide every crucial steps of my thesis, but he also provided, supported and suggested other aspects of this paper. I really appreciate his supervision since he provided very detailed instructions, though he was extremely busy.

I would also like to give thanks to my former teacher: Dick Stenmark at IT Göteborg department for helping me with the methodology research methods. Moreover, I would like to thank Professor Giuliano Antoniol and Professor Alexander Egyed at Center of Excellence for Software Traceability for their preliminary comments on this research topic.

Last but not least, I am deeply grateful to my family and friends for their support, encouragement and always being with me through the challenges that I faced.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The concept 'traceability' is a very broad and general term. Traceability is an attribute of any artifact in a software system. In [58], traceability is referred as the potential for traces to be set up and used. It consists of three concepts: Requirement traceability (RT), Software traceability and Systems traceability. However, RT is the most common concept and seems to be mentioned in most literature discussing issues related to traceability of a software system.

Different researchers have their own written words for the definitions of traceability. However, the definition made by Gotel and Finkelstein [59] can probably be considered as the most comprehensive and is referred to by many researchers. According to Gotel and Finkelstein, *"Requirement traceability refers to the ability to describe and follow the life of a requirement in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)"*.

Traceability activities play very important roles in a software project. RT is considered as an index of software quality [133], [169]. Traceability is one of the recommended activities for the system requirement specification that CMMI [174] and ISO 15504 [170] consider as "best practice and strongly suggest its usage" because traceability allows various stakeholders of a system to understand the different relationships among produced artifacts during the product development process [140]. It could be, therefore, concluded that traceability is a very important task no matter what software development approach is used.

Looking first glance at existing traceability methods, it could be said that many of traceability approaches seem to be specifically designed for traditional software development processes, e.g. Waterfall. They have major shortcomings. These techniques seem to be characterized with traditional software development processes [61], [51], [22] so that they are drawback and cannot adequately support other software development methods such as: Scrum, XP, Lean, Streamline, Crystal and Kanban (hereafter called Agile methods). Researchers in [79], [162], [52] said that these current characterized traceability approaches are heavy, with high overhead, increase administrative work, thus, making the process longer.

Shortcomings of current traceability techniques mentioned above will cause practitioners in Agile software environment to face difficulties when adding traceability to Agile software projects. For example, there could be lack of traceability methods specifically designed for Agile software processes and may cause a dilemma for software development organizations [97], [79], [60]. This is confirmed since the thesis author conducted the pilot search and found very few papers discussing traceability in Agile. In addition, some researchers stated that traditional traceability methods could not be applied to Agile software environment [22], [129], [17] because they could cause detrimental or downside impacts to Agile projects.

From the problems and shortcomings of many current traceability approaches mentioned above, together with the reality that Agile software development methods have been more and more adopted in the industry [79], [46], [65], it could be said that more and more literature are really needed to discuss the traceability in Agile software development. That is also the main reason for the author to conduct this thesis paper, which has aims and objectives stated in the next coming subsection.

1.1 Aims and Objectives

The aim of this thesis paper is to investigate how traceability is implemented in Agile software environment by:

- Identifying important types of traceability in Agile software development.
- Identifying traceability mechanisms that Agile companies are using, explaining why other traceability mechanisms are not used.
- Summarizing feedbacks evaluating how these traceability mechanisms work well in Agile companies.
- Identifying benefits when implementing traceability in Agile projects.
- Discovering challenges when conducting traceability in Agile software projects.

1.2 Research Questions

The research topic was divided into some research questions which are illustrated in the Table 1.1:

Table 1.1: The research questions and their aims

Research Question	**Aim**
RQ1:What types of traceability are important for software development in organizations subscribing to being Agile?	To gather the list of traceability types and point out which types are important based on the results from the interviews conducted in industry (see more details in the section 7.2.1)
RQ2: What mechanisms are used by Agile companies to achieve traceability?	To know mechanisms (tools, methods, frameworks, approaches, etc) that Agile companies are using to implement traceability.
RQ3: What mechanisms work well in Agile development organizations?	To gather opinions of practitioners evaluating if traceability mechanisms work well in Agile development organization.
RQ4: What challenges exist for realizing traceability in Agile organizations?	To gather opinions, feedbacks from the industry and literature identifying difficulties when conducting traceability in Agile organizations.
RQ5: What benefits exist in realizing traceability in Agile organizations?	To gather opinions, evaluations from the industry and literature identifying benefits when conducting traceability in Agile organizations.

1.3 Expected Outcomes

This thesis paper aims to identify some outcomes. First of all, this study will discuss why some types of traceability are *important* and why other types are *not* important in Agile projects. Second, the thesis will summarize the number of traceability mechanisms that Agile companies are using. Then, these traceability mechanisms will be evaluated by practitioners to see if they work well in Agile software projects. Finally, some challenges and benefits of implementing traceability in Agile projects will also be discussed.

1.4 Terminologies

Some terms frequently used in this thesis are described in the Table 1.2:

Table 1.2: Definitions for terms used in this study

Terms	**Definitions**
Traceability	An attribute of any artifact in a software system. Traceability is referred as the potential for traces to be set up and used. It consists of three concepts: Requirement traceability (RT), Software traceability and Systems traceability. However, RT is the most common term and seems to be mentioned in almost literature discussing issues related to traceability in a software system.
Tracing links/relation	a relation or link between two software artifacts or elements.
Software element or artifact	It can be a requirement, UML design diagram, source code class and etc
Types of traceability	A specific type, which is named for a tracing link/relation. Example: dependency, contribute, rationale, etc (see more details in the section 2.1.7). In this paper, important types will be based on findings extracted from the industry interviews and literature (see in the section 7.2.1).
Traceability mechanism	A collection word referring to tools, approaches, methods, techniques, etc which help to achieve traceability. This thesis will collect feedback of practitioners, who will evaluate how a specific mechanism work well in Agile development organizations.
Agile practice	Agile development process could have many practices. These practices (see more details in the section 2.2.2) could be considered as tools that help practitioners to achieve main principles, which are mentioned in the section 2.2.1.

Chapter 2

Background

2.1 Overview of traceability

2.1.1 Definition of traceability

Software traceability refers to the ability to relate software artifacts created throughout the development life-cycle of a software system. Over the last years, the software and system engineering communities have developed a large number of approaches to address various aspects of traceability. Among published papers, many researches focused on the study of the traceability definition.

The most dominant definition of traceability made by Gotel and Finkelstein [59] mentioned above describes both pre-requirement specification (RS) and post-RS traceability. The former describes all aspects of requirements before documenting them in requirements specification while the latter refers to all aspects of requirements after documenting them in requirement specification. For this reason, this definition has been referred by various literature discussing the traceability topic.

Apart from the definition of Gotel and Finkelstein, some other researchers had different words describing traceability. Edwards and Howell [53] defined traceability as a technique used to "provide a relationship between requirements, designs and final implementation of the system", meaning that a tracing relation occurs among three major software artifacts: requirements, design and implementation or coding. In other words, this definition just focuses on describing "Forward-From-Requirement" traceability.

Palmer [117] stated that "traceability gives essential assistance in understanding the relationships that exist within and across software requirements, design and implementation". This definition just describes aspects of post-RS traceability. Ramesh & Jarke [133] concurred with Palmer when defining RT as "the ability to relate requirements specifications with other artifacts created in the development life-cycle of a software system". This definition focused on reflecting tracing links from requirements to their subsequent software artifacts in the development life cycle. Murray et al [16] also described aspects of post-RS traceability when defining traceability as "the ability to identify requirements at different levels of abstraction and to show that they have been implemented and tested".

Traceability issues also draw attention from international standards and organizations, for example: the ANSI/EEE standard 830-1998 defined traceability as: "a software requirements specification is traceable if

(i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation" [21],[130],[85],[16]. The US department of defense standard DoD Std2167A defined: "traceability means that the document in question is in agreement with a predecessor document, to which it has a hierarchical relationship. Traceability has five elements:

- The document in question contains all applicable stipulations of the predecessor document.
- A given term, acronym, or abbreviation means the same thing in all documents.
- A given item or concept is referred to by the same name or description in the documents.
- All material in the successor document has its basis in the predecessor document, that is, no untraceable material has been introduced,
- The two documents do not contradict one another" [85].

2.1.2 The importance of traceability

Traceability of software artifacts is considered as an important factor in supporting various activities in the development process of a software system [143]. In general, the objective of traceability is to improve the quality of software systems. More specifically, traceability information can be used to support some activities such as: the change impact analysis, software maintenance and evolution, the reuse of software artifacts by identifying and comparing requirements of the new system with those of the existing system.

Large-scale industrial projects often comprise of thousands of software development artifacts, for example: requirements documents, design documents, code, bug reports, test cases, and etc. The goal of software traceability is to discover relationships between these artifacts to facilitate the efficient retrieval of relevant information, which is necessary for many software engineering tasks [6].

Traceability helps developers to control and manage the development and evolution of a software system. It has been defined as the "ability to follow the life of a requirement in both a forward and backward direction" in order to understand the origins of the requirement and also to determine how a requirement has been realized in downstream work products such as design, code, and test cases [59].

2.1.3 Purposes of traceability

Implementing traceability helps to conduct important tasks, that is to evaluate how changes in an element may impact other parts of the system or the change impact analysis. Patrik and Cleland-Huang [102] stated that change impact and coverage analysis are major and common tasks of implementing traceability and these tasks can be achieved by basing on traceability links that users create when using requirement management tools.

Change impact analysis is a fundamental software engineering task, which helps to determine potential effects upon a system resulting from changes in requirements of a system. Jessica et al [44] achieved this task by developing a traceability technique mainly basing on traceability-based algorithm. In their literature called "Impact Analysis from multiple Perspectives: Evaluation of traceability Techniques", Salma Imtiaz et al [75] considered change impact analysis as a core activity of Requirement Change Management process and confirmed that traceability information is the pre-requisite to achieve the change impact analysis activity.

To design, implement and maintain a large software system, conducting traceability is the key to coping with change, which occurs very often throughout the development evolution process. The main purpose of traceability is to discover how and where change affects software artifacts at different stages [78]. Pedo Sanchez et al [142] shared their ideas when saying that changes to requirements occur very often during the whole development process and traceability reports can assist analysts to evaluate the impact of these changes before implementing them.

The second purpose of traceability is to help the validation of requirements. Stefan and Jens [158] summarized extensive literature and stated that most traceability tools or methods concentrate on such software engineering tasks as change impact analysis, validation and verification. These authors elaborated that tracing information is specially valuable to such software engineering activities as: validation and verification, which help to identify pairs of linked artifacts. Traceability reports are used to validate if all requirements are supported. In [142], specifically, researchers checked that all requirements are represented by means of the model elements obtained according to the Domain Specific Language (DSL) and traceability reports could be filtered to find those DSL elements that are linked to the analyzed requirement.

Software maintenance is one of the core goals of implementing traceability in a software system. Spanoudakis and Zisman stated that based on traceable semantics, traceability links can give information that can be used in various ways throughout the development life cycle [143]. It was written in [143] that traceability relationships may be used to support the assessment of the implications of the changes in a system and the effective execution an integration of such change during the development, maintenance and evolution of a system.

Traceability provides stakeholders of a system with means to maintain system design rationales, showing when the system is complete and setting up the change control and maintenance mechanisms [131]. Traceability information could be very important during the whole life-cycle maintenance and on the development of similar systems. The primary goal of software traceability is to discover relevant links among software artifacts and these links are very helpful towards software maintenance activities. Particularly, traceability is an essential issue when we design and maintain huge software systems, which are prone to change [78].

Stefan and Jens [158] wrote that traceability information is crucial for implementing software maintenance since traceable information can help us to understand the system. Traceability was seen to provide the insurance and if software companies do not sufficiently invest in traceability, these companies will have to pay higher costs for software maintenance task. That seems to mean traceability and software maintenance are integral parts.

Finding and keeping tracking relationships among software artifacts is also one of the main purposes of conducting traceability. This will make it easy for the efficient retrieval of related information [6], which is necessary for many software engineering activities. Edwards and Howell [53] wrote that traceability can show relationships among artifacts for team members to conduct their specific tasks. For instance: discovering relationships allows designers to demonstrate that their design components have accommodated requirements and to make it easy for the recognition of requirements, which have not been met yet by relevant design elements.

Traceability can enable the linking of different software artifacts and make it explicit that a software system has been implemented to accommodate its requirements. Accurate traceability information can create correct relationships among artifacts [163]. As the result of this, we could ensure that the implementation is verified to satisfy the software requirement description. Traceability links help stakeholders to understand the dependencies that exist among elements or artifacts of a system and these dependencies allow stakeholders to perform important software engineering tasks such as the change impact analysis [118].

Implementing traceability activities aims at verifying and debugging the system. The majority of traceability tools or methods focus aspects of post traceability and mainly reflect requirement refinement, requirement allocation and compliance verification [87]. For example, low-end users create traceability links to mock-up or make a representation for the dependency of requirements, allocation of requirements to system components and compliance verification [133]. Verifying a system before handing in to its customer is a must-have task. Specifically, at the testing phase, the development team will use traceability to verify and prove to the customers that the system meets specified requirements and the project is complete [131].

Hazeline and Richard [7] stated that once software traceability is fully realized, its information is very effective to system comprehension and system debugging. For instance: a quality assurance manager can obtain system verification by tracing test cases and test reports back to requirements- "Backward-To-Requirement" traceability. Min Deng et al [43] developed a traceability tool to focus on supporting verification and validation of UML models and formalizations.

To reuse software artifacts is also one of the main goals of performing traceability in software projects. We can see the reuse issue in software product line in industries, e.g making software for various types of car or telephone. To save resources, the development team is trying to reuse as much requirements as possible between variants [87]. Giuliano Antoniol et al [5] wrote that "Reusable assets from existing software systems has emerged as a winning approach to promote the practice of reuse in industry". For example, traceability links from code to text documents can help to find reusable candidate components.

In software engineering community, researchers widely agree that traceability links can surely help to discover reusable software artifacts at different levels [143]. "Forward From Requirement" traceability means tracing from requirement specification documents to design artifacts, e.g. UML design diagram and from design artifacts to source code. Software engineers follow these tracing links to locate concrete reusable artifacts in an application frame at low levels of abstraction [30]. For example, after evaluating and analyzing the similarity for requirements of a new system, a team member decides to reuse a specific requirement of a group of requirements of the exiting system to put in the list of requirements of this new system.

Apart from major purposes of traceability elaborately discussed above, performing traceability can gain other goals such as: change management, process compliance and organizational learning. George and Zisman [143] said that tracing links help the development team to make decisions about whether or not such changes should be implemented and with which priority. Software engineers need to achieve process compliance to ensure that procedures, tasks, etc have been conducted, e.g code review [23]. Working with Agile development methods is an effective way to spread knowledge or experience among team members. Performing traceability in an Agile software project helps to transfer knowledge. For example, experienced members document rationales behind critical decisions with the aim of helping new team members get understandings of the system [23].

2.1.4 Traceability directions

As mentioned above, traceability refers to the ability to associate software artifacts. However, there are a lot of traceability relationships, which need to be categorized and grouped according to reasonable, logical and meaningful manners. In software engineering, researchers have used different ways to classify traceability directions. According to the investigation of Gotel and Finkelstein [59], there are two basic traceability directions: pre-RS and post-RS. Pre-RS traceability refers to aspects of a requirement's life prior to its inclusion in the RS (requirement production). Post-RS traceability is related to aspects of a requirement's life and these aspects stem from its inclusion in the RS (requirement deployment). These two directions are illustrated in the Figure 2.1:

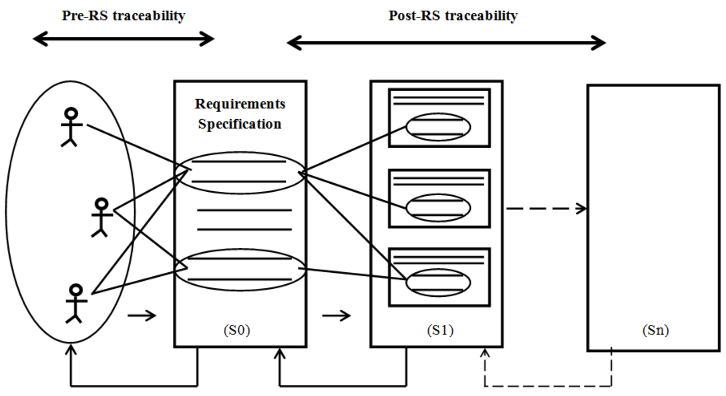


Figure 2.1: Pre-RS and Post-RS traceability

Gotel and Finkelstein wrote: though forward and backward requirement traceability are explicitly essential, the emphasis needs to be put on pre/post RS traceability directions since problems related to requirement traceability were found to concentrate on the current lack of distinction in practice.

According to Alan Mark Davis [42], traceability can be categorized into four directions as followed:

- Forward from requirements traceability: responsibility for requirements achievement must be assigned to system components, such that accountability is established and the impact of requirements change can be evaluated.
- Backward to requirements traceability: compliance of the system with requirements must be verified, and go-plating (designs for which no requirements exist) must be avoided.
- Forward to requirements traceability: changes in stakeholders needs, as well as in technical assumptions, may require a radical reassessment of requirements relevance.
- Backward from requirement traceability: the contribution structures underlying requirements are crucial in validating requirements, especially in highly political settings.

Analyzing the descriptions of four traceability directions made by Alan M.Davis, Matthias Jarke [80] stated the first two are actually described in post-RS traceability and the latter two enable pre-RS traceability mentioned in [59].

In the literature named "Engineering and Managing Software Requirements" [8], A. Arum and C. Wohlin mentioned that pre-RS traceability refers to aspects of a requirement's life from the point, at which this requirement is not included in the requirement specification. In pre-RS traceability, requirements are linked to their origin and other requirements. In backward traceability direction, Tony Gorschek [62] defined the origin of requirements as a person or a group of people, or a document. Arum and Wohlin [8] made the representation for four directions of traceability in the Figure 2.2:

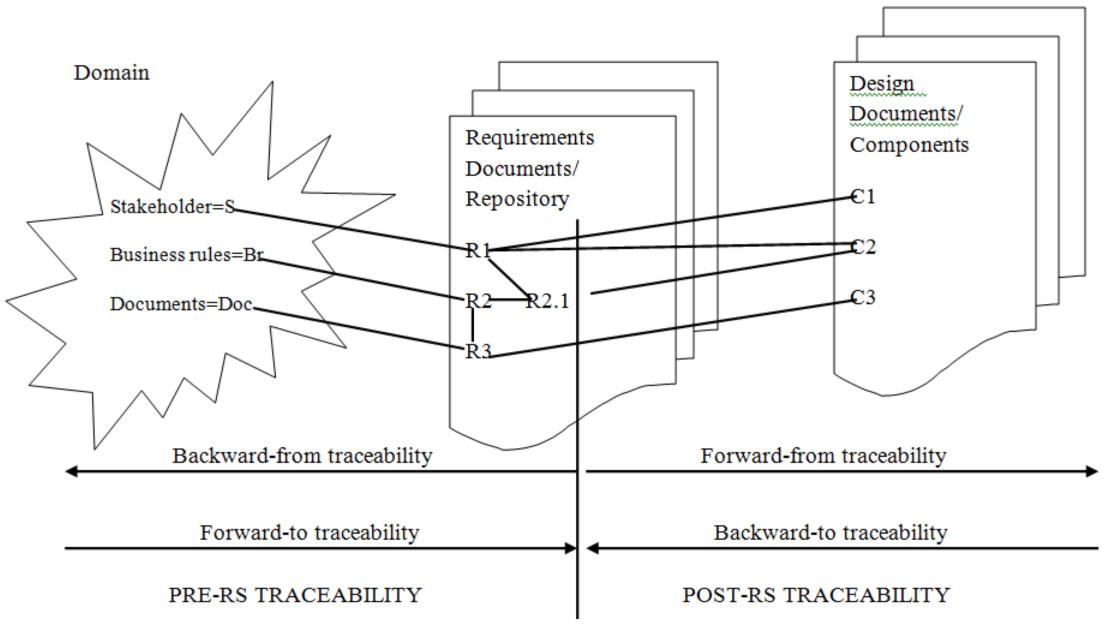


Figure 2.2: Four directions of Traceability

Looking at the figure 2.2, we can see that pre-RS traceability includes "Backward-from" and "Forward-to" traceability directions. Requirement R2.1 is linked to two requirements R1 and R2. On the contrary, post-RS traceability refers to aspects of a requirement's life from the point, at which this requirement is already included in the requirement specification and forward. Post-RS traceability covers "Forward-from" and "Backward-to" traceability directions. In figure 2.2, three requirements R1, R2.1 and R3 are linked back and forth in respective to three design components C1, C2 and C3.

The book named "Software and Systems Traceability" [24] summarized different ways of classifying traceability directions mentioned before. Two traceability directions were supplemented: Vertical and Horizontal traceability. The former is commonly used when we trace software artifacts at their different levels of abstraction so as to accommodate life cycle-wide or end-to-end traceability, for example: tracing a link from a requirement to a source code module. This direction of traceability may employ both forward and backward tracing relations. The latter is usually used when tracing artifacts at same the level of abstraction. For instance: tracing links between all requirements made by a team member; or tracing relations between requirements that are concerned with a particular non-functional requirement; or tracking a requirement specified at different versions of requirement documents. Horizontal traceability may also use both forward and backward tracing directions, like the Vertical direction,.

2.1.5 Traceability Mechanisms

Mechanisms is a word collection, which refers to a tool, method, framework, approach or technique, etc. A traceability mechanism is to help us to achieve traceability. Traceability mechanisms can be classified into three different types: manual, semi-automated and automated traceability tools. Looking at existing traceability approaches, we recognize that many of these tools require users to identify, create or maintain traceability links manually. We can see manual tools or approaches described in extensive literature such as:

[15, 30, 119, 58, 45, 177, 173, 178, 179, 86]. Creating traceability links by manually is difficult, error-prone and time consuming. According to George and Zisman [143], manual declaration of traceability relations is normally supported by visualization and displayed tool components, at which documents will be tracked. Following these tool components, it may make it easier for users to link elements described in these documents.

To alleviate and tackle difficulties of manual tools, some other approaches were introduced to support semi-automated traceability as we can see in literature [59, 25, 26, 54, 55, 120, 119]. George and Zisman [143] grouped semi-automated tools into two groups. The pre-defined link group means that traceability approaches have to depend on previous-defined links, basing on which new tracing relations will be generated. The other group named "process-driven group" refers to traceability techniques supporting to generate traceability links as a result of the software development.

Over the past decades, many approaches have been proposed to support automated generation of traceability relations. Some approaches [5, 66, 103] use information retrieval (IR) techniques, which compare the similarity between the text in software artifacts. The higher the textual similarity between two artifacts, the higher the likelihood that a link exists between these two artifacts [24]. Other frameworks [144, 165, 166] follow the rule-based traceability, which develops XML-based traceability rules, basing on which traceability relations between artifacts such as: requirement statements, use cases and object models are automatically generated. In addition, automatic tools are based on integrators as mentioned in [145]. This automated tool uses special integrators, which can identify and create tracing links between new artifacts and previous defined relations.

2.1.6 Tracing links/relations

In a software system, people participating in developing the system are called Stakeholders, who can be a customer, product owner, tester, project manager, etc. A stakeholder can make contribution to making one or several elements or artifacts of the system. Artifacts of a software system can be: a requirement, a rationale behind the generation of a design component, or a test report, etc. Basically, software artifacts can be put into four main groups [143] as followed:

- Artifacts related and belonging to requirement documents can be named as requirements.
- Artifacts related and belonging to design documents can be named as designs
- Artifacts related and belonging to source code can be named as codes
- The rest of artifacts are named "others", which include: test cases, test reports, goal documentation, technical manual, etc.

A specified tracing link starts from the source artifact and comes to the target artifact. Tracing relations can be divided into two types [24]. The first one is called "primary tracing link", when a link is traversed from its specific source artifact to its specific target artifact. The other is named "reverse tracing link", when a link is traversed from its specified target artifact back to its associated source artifact.

Links from requirements to other artifacts could be:

- Requirement-To-Requirement: tracing from a requirement to another requirement and this link has been discussed in [16, 2, 15, 30, 59, 104, 121, 119, 2, 87, 166]

- Requirement-To-Design: tracing from a requirement to a design component and this link has been discussed in [133, 30, 31, 8, 22, 42]
- Requirement-To-Code: tracing from a requirement to a code module and this link has been discussed in [133, 54, 105, 5, 42]
- Requirement-To-Others: Tracing from a requirement to any other artifacts that do not belong to requirements, design, or code. This relation has been discussed in [96, 133, 121, 104, 119]

Links from Design to other artifacts could be:

- Design-To-Requirement: tracing from a design component back to a requirement and this link has been discussed in [42, 172]
- Design-To-Design: tracing from a design element to another design element and this link has been discussed in [96, 133, 7, 54, 167]
- Design-To-Code: tracing from a design element to a code module or element and this link has been mentioned in [16, 30, 15, 121, 172]
- Design-To-Others: tracing from a design to any other artifacts that do not belong to requirements, design, or code. This relation has been discussed in [96, 133]

Links from implementation to other artifacts could be:

- Code-To-Requirement: tracing from a code element or module back to a requirement and this link has been described in [42, 172]
- Code-To-Design: tracing from a code element back to a design component and this link has been described in [4]
- Code-To-Code: tracing from a code element to another code element and this link has been covered in [133, 16]
- Code-To-Others: tracing from a code to any other artifact that do not belong to requirements, design, or code. This relation has been discussed in [133, 105]

Apart from tracing links starting from three main source artifacts to relevant target artifacts, we may need to think of two other types of links. First, others artifacts can be traced to each other. For example a relationship from a test case to a test report and this link has been mentioned in [133, 105]. In addition, when implementing traceability, we pay attention to stakeholders, who make software artifacts or elements and this tracing relation has been discussed in [96, 143, 16, 132, 22].

2.1.7 Types of tracing links/relations

A tracing link may be described by one or more than one types of relationship, depending on concrete situations. For example: suppose that we have two elements of a system named e1 and e2 (an element can be: stakeholders/requirement/design component/code/test data/ etc). e1 and e2 can be two totally different requirements and depend on each other, so the their relationship is of dependency. Or e1 will be modified and replaced by e2 in a new requirement document and this relation is 'revolution'. This study followed eight types of relationship defined in [143]:

- *Dependency: e1 depends on e2 if the existence of e1 relies on the existence of e2. Example: dependency between 02 requirements, or dependency on one requirement and one design component.*
- *Refinement: this type is used to identify how a complex element can be broken; or how elements of a system can be combined to form other elements; or how an element can be refined by another element.*

- *Evolution: e1 evolves to e2 if e1 has been replaced by e2 during the development lifecycle of the system. Example: Req1 in the requirement document V.01 has been replaced by Req1.1 in the requirement document V.02*
- *Satisfaction: e1 satisfies e2 if e1 meets the expectation or needs of e2. Example: e2 (a design component) is used to make sure that e1 (a requirement) is satisfied by the system.*
- *Overlap: e1 overlaps with e2 if both e1 and e2 refer to a common feature of a system. Example: e1 (a requirement statement) and e2 (use-case) both refer to a common feature of the system.*
- *Conflict: this type shows the conflict between e1 and e2. Example: e1 and e2 are two requirements and they are in conflict with each other.*
- *Rationale: this type is used to represent and maintain the rationale behind the creation and evolutions of an element.*
- *Contribute: this type is use to point out who is the author to make an artifact.*

2.1.8 The traceability map

To help readers to have an overview of software traceability, it is necessary to gather information and issues related to traceability on one map. This map is the foundation, on which the industry interviews and literature review were conducted and the map summarizes previous sections and could be illustrated as followed:

BFT: it links reqs to its origin (people, docs, rationale, etc)

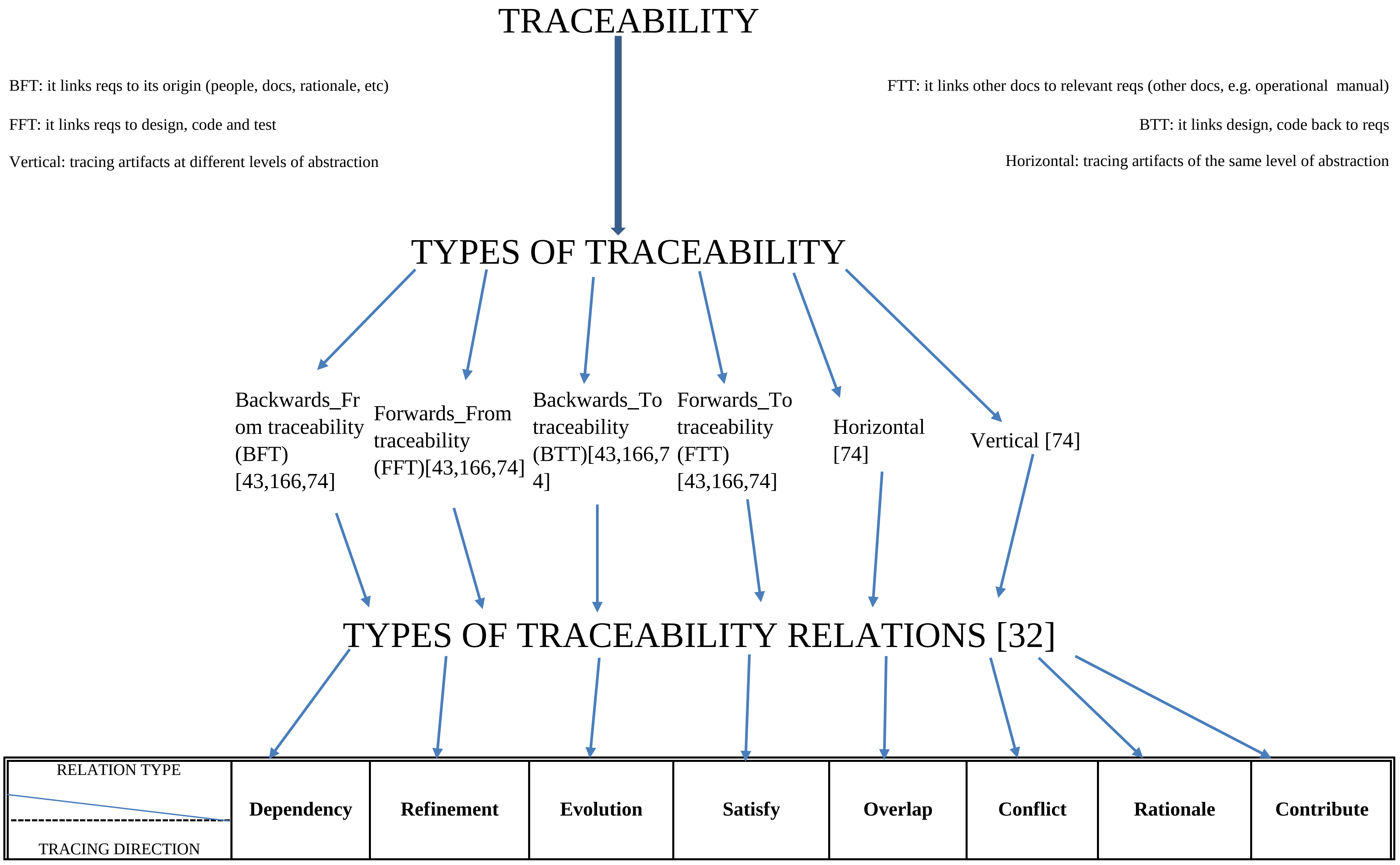
FFT: it links reqs to design, code and test

Vertical: tracing artifacts at different levels of abstraction

FTT: it links other docs to relevant reqs (other docs, e.g. operational manual)

BTT: it links design, code back to reqs

Horizontal: tracing artifacts of the same level of abstraction


TRACEABILITY
TYPES OF TRACEABILITY
Backwards_From traceability (BFT) [43,166,74]
Forwards_From traceability (FFT)[43,166,74]
Backwards_To traceability (BTT)[43,166,74]
Forwards_To traceability (FTT) [43,166,74]
Horizontal [74]
Vertical [74]
TYPES OF TRACEABILITY RELATIONS [32]
RELATION TYPE
TRACING DIRECTION
Dependency
Refinement
Evolution
Satisfy
Overlap
Conflict
Rationale
Contribute

TYPES OF TRACEABILITY RELATIONS [32]

RELATION TYPE / TRACING LINK	**Dependency**	**Refinement**	**Evolution**	**Satisfy**	**Overlap**	**Conflict**	**Rationale**	**Contribute**
Requirement_To_Requirement [25, 57, 48, 41, 2, 69, 159, 49, 65, 37, 67]	Alexander [57]	Gotel et al [163]	SIB[41]	Alexander [57]	Bayer et al[48]	Alexander [57]	Letelier[73]	
	Gotel et al [163]	Letelier[73]	Gotel et al [163]	TOOR[159]	Egyed[60]	PRO-ART[49]	REMAP[169]	
	PRO-ART[49]	Mohan et al[69]	TOOR[159]	PRO-ART[49]	Gotel et al [163]	Ramesh et al [3]		
	Ramesh et al [3]	TOOR[159]	REMAP[169]	Dick[51]	Rule-base tracer [65, 67]			
	Kenethen et al [37]	REMAP[169]	PRO-ART[49]					
	Rule-base tracer [67]	PRO-ART[49]						
		Kenethen et al [37]						
Requirement_To_Design [3, 41, 71, 45, 10, 43, 166, 74]	SIB[41]	Letelier[73]	REMAP[169]	Ramesh et al [3]	Cysneiros et al[71]	Kozlenkov & Zisman[170]	Letelier[73]	
	Egyed[60]	REMAP[169]		CORE[52]	Egyed[60]	Ramesh et al [3]	REMAP[169]	
	Ramesh et al [3]							
Requirement_To_Code [3,	Egyed[60]	Maletic et al[72]	Ramesh et al [3]	Antoniol[40]				

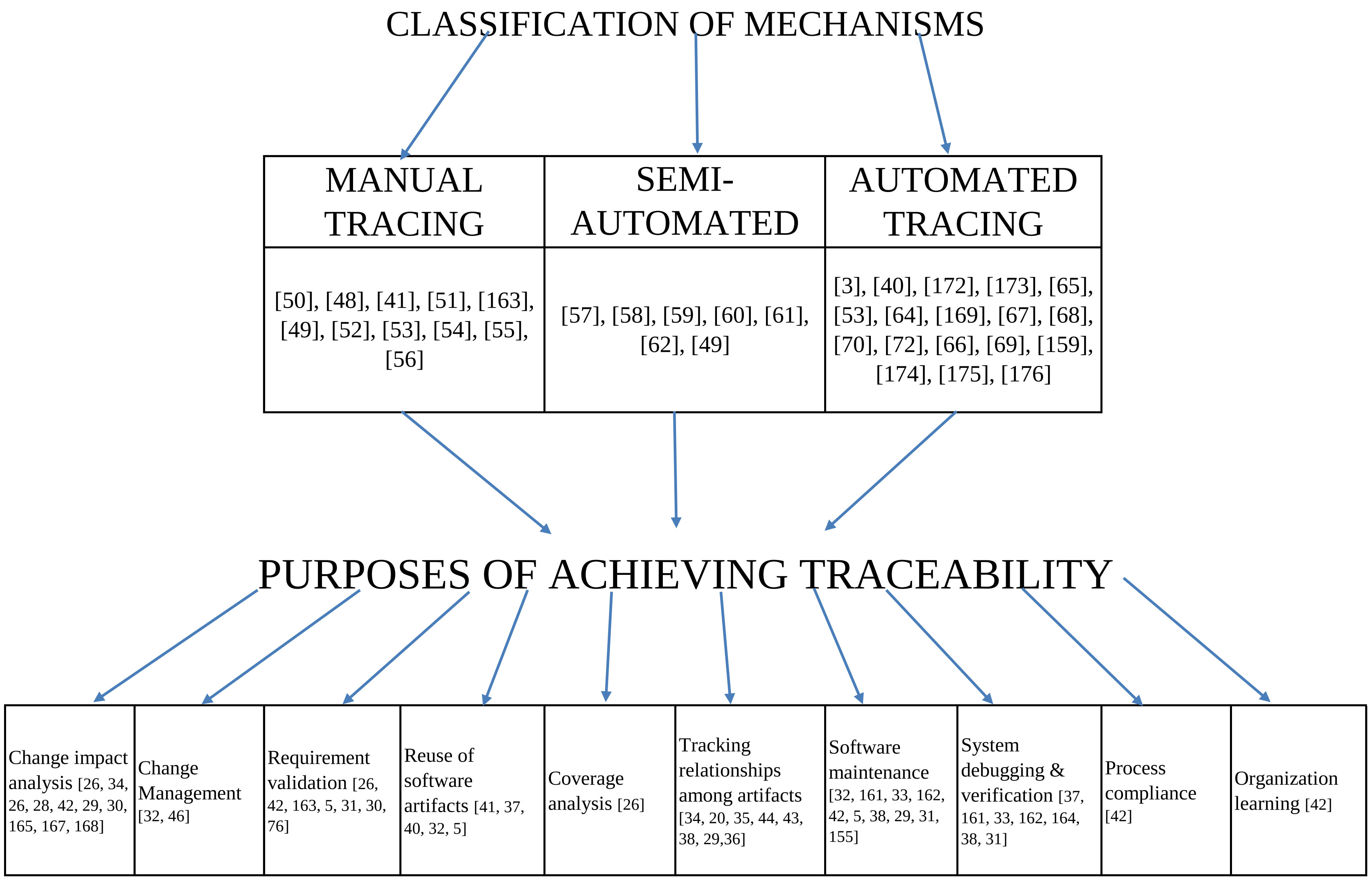
RELATION TYPE / TRACING LINK	Dependency	Refinement	Evolution	Satisfy	Overlap	Conflict	Rationale	Contribute
60, 72, 40, 43, 166, 74]	Maletic et al[72]							
Requirement_To_Other [1,		Letelier[73]		Dick[51]	Cysneiros et al[71]	Ramesh et al [3]	Letelier[73]	
		Mohan et al[69]		Ramesh et al [3]	PRO-ART[49]			
		TOOR[59]		TOOR[159]				
Design_To_Requirement [43, 166, 74]								
Design_To_Design [73, 3, 37, 60, 75]	Egyed[60]	Kenethen et al [37]		Ramesh et al [3]		Xlinkit[171]		
	Kenethen et al [37]					Zisman et al[75]		
						Ramesh et al [3]		
Design_To_Code [25, 41, 48, 72, 74]	SIB[41]	Letelier[73]	SIB[4]	Ramesh et al [3]			Letelier[73]	
	Egyed[60]		Maletic et al[72]					
	Maletic et al[72]							
Design_To_Other [73, 3]		Letelier[73]		Ramesh et al [3]		Ramesh et al [3]	Letelier[73]	
Code_To_Requirement [43, 166, 74]								
Code_To_Design [74]								

RELATION TYPE / TRACING LINK	**Dependency**	**Refinement**	**Evolution**	**Satisfy**	**Overlap**	**Conflict**	**Rationale**	**Contribute**
Code_To_Code [3, 25]				Ramesh et al [3]				
Code_To_Other [3, 72]	Maletic et al[72]		Maletic et al[72]	Ramesh et al [3]				
Other_To_Other [73, 3, 67, 160]	Xu et al [160]	Letelier[73]		Ramesh et al [3]	Rule-base tracer [67]		Letelier[73]	
		Xu et al [160]						
Stakeholders_To_Requirement								Gotel et al [163]
Stakeholders_To_Design								
Stakeholders_To_Code								
Stakeholders_To_Other								

MECHANISMS TO ACHIEVE TRACEABILITY [32]

Mechanisms (methods/approaches/practices/frameworks/techniques/tools)

CLASSIFICATION OF MECHANISMS


CLASSIFICATION OF MECHANISMS
MANUAL TRACING
SEMI-AUTOMATED
AUTOMATED TRACING
[50], [48], [41], [51], [163], [49], [52], [53], [54], [55], [56]
[57], [58], [59], [60], [61], [62], [49]
[3], [40], [172], [173], [65], [53], [64], [169], [67], [68], [70], [72], [66], [69], [159], [174], [175], [176]
PURPOSES OF ACHIEVING TRACEABILITY
Change impact analysis [26, 34, 26, 28, 42, 29, 30, 165, 167, 168]
Change Management [32, 46]
Requirement validation [26, 42, 163, 5, 31, 30, 76]
Reuse of software artifacts [41, 37, 40, 32, 5]
Coverage analysis [26]
Tracking relationships among artifacts [34, 20, 35, 44, 43, 38, 29,36]
Software maintenance [32, 161, 33, 162, 42, 5, 38, 29, 31, 155]
System debugging & verification [37, 161, 33, 162, 164, 38, 31]
Process compliance [42]
Organization learning [42]

2.2 Overview of Agile software development

In february, 2001 the group of seventeen people met and discussed to introduce the Agile concept. After this meeting, the outcome was Agile Software Development Manifesto. More details of Agile can be seen at [180]. The thesis author just focused on summarizing Agile in two subsections below:

Principles of Agile

Agile has twelve principles behind the Agile Manifesto as mentioned in [180]. These principles are summarized in the following order:

1. The highest priority is to satisfy customers by delivering early and continuously valuable software products.
2. Welcoming changes at requirements, even at late phases of the development process.
3. Delivering workable software products frequently in a wide rage of time, e.g from a couple of weeks to a couple of months.
4. Business people and developers must work together daily throughout the project.
5. Giving the need, supporting and trusting team members with the aim of motivating them to get the project done.
6. Face-To-Face communication is the best way to convey information within the team.
7. Producing workable software products is the main criteria for checking the project progress.
8. Agile processes promote sustainable development.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done, is crucial
11. Having self-organizing teams to get best architectures, requirement and designs.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

2.2.1 Practices of Agile

Practices are categorized into four different groups: Extreme programming, Scrum, Lean software development and Feature-driven development as illustrated in the Agile map (see more details in the section 2.2.3). In this thesis, 42 practices and their relevant definitions were extracted, summarized from papers [23, 81, 12, 153, 83] and shown in the Table 2.1:

Table 2.1: The list of Agile practices and their definition

N.o	Practice name	Definition/description
1	Pair Programming	Two developers work in pairs, one as the driver to write code and the other as navigator to review code.
2	Test-Driven Development	A developer writes a number of automated unit test cases. He will run these test cases to make sure they will get failure. Then, he writes codes to make these test cases get passed.
3	Coding standards	This practice shows the adaptation of common coding conventions during whole development process.

4	On-site customer	Customers sit together with the development team and actually get involve in developing the software system.
5	Coding standards	That means all code in the system look as if they were written by a single, very competent individual.
6	System metaphor	This practice shows the process to substitute an existing architecture with the new one.
7	Small release	This practice shows the process to get feedback from users so early and focus on the effective growth of software in increments.
8	Collective code ownership	This practice shows the code will not be the proprietary of single person.
9	Sustainable pace	This practice shows how a developer can work with efficiency after it had become tired.
10	Planning game	A planning section, where story cards are defined and prioritized together with a customer.
11	Refactoring	This practice shows the improvement by simplifying the code to make it cleaner without changing its functionalities.
12	Continuous integration	This practice shows that the integration testing will be applied each time when a new build is developed and integrated into the existing system.
13	40 hour week	A working culture where work is limited to working hours to increase creativity, health and avoid overtime.
14	Acceptance test	An automated acceptance test that is run with past or fail results, which are defined by a customer.
15	Common room	A working environment where the whole team is working as close as possible preferably in one room.
16	Product Backlog	It is an ordered list of requirements that are maintained for a product. It contains Product Backlog items that are ordered by the product owner.
17	Sprint planning	A planning activity that consists of two meetings. First, stakeholders refine and prioritize the product backlog. Second, team and product owners plan how to achieve iterative results and create task lists.
18	Stand-up meeting	A short 15-20 minutes daily meeting where each team member answers 03 main questions: What is done so far; What is planned to be done until the next meeting; What are obstacles to achieve iterative goals.
19	Daily build	The work in print is divided into daily blocks that leads to daily builds.
20	Sprint review	A meeting where a review for a Sprint is executed and a demo of a product is presented at the end of that Sprint.
21	Scrum master firewall	Scrum Master (a manager) assures that work in the team is happening and no undesired activities exist (extra work added to a Sprint or any outside interruption) within the team.
22	Lock priority within Sprint	Priorities chosen at the beginning of a Sprint. No extra work that could be added to iteration is to be related to maintain team focus on the goal. In case extra work is added, some work should be removed.
23	Decision in one hour	It refers to the decision making process that does not take longer than one hour. No decision is worse than a bad decision and a bad decision can be reversed.
24	High level design	The sketch of design only gets basic understandings about the system.
25	Self-organizing team	The culture where the team has authority and resource to choose the best way to achieve Sprint goals, to prioritize work and to solve its own problems.
26	Team of seven	The teams that consists of no more than seven people to assure efficiency and smooth communications.
27	Eliminate waste	To remove everything that does not create a clear value for a customer (product).

28	Built quality in	That focus on eliminating defects as soon as they are detected; to avoid creating defects in the first place.
29	Empower the team	To develop an organization where each person has an authority to prioritize, take responsibility and come up with solutions instead of having someone telling what to do and how to do it.
30	Create knowledge	It encourages systematic learning throughout the development cycle and makes sure that tacit knowledge is shared.
31	Deliver as fast as possible	Only core functionality of the most valuable business processes is implemented in the first iteration to earn value form early on and to get feedback right away. Changes and new functionalities are added on demand (i.e. just-in-time).
32	Optimize the whole	Optimizing the whole value stream from the time it receives an order to address a customer need until software is deployed and the need is addressed; avoid sub-optimization.
33	Decide as late as possible	To decide details as late as possible to minimize the cost of change so the team can try many alternate solutions.
34	Amplify learning	The business process management deals with wicked problems that cannot be planned in every detail; the right the first time approach does not work, but the solution has to be continuously improved selected.
35	Build integrity in	It is best achieved by close collaboration between work-flow modelers and work-flow users during build-time and by allowing modifications during run-time by the user. Work-flow model is modified and tested directly by the user (customer test). the work-flow modeler should create and run corresponding developer tests.
36	Domain object model	It refers to a process of creating the framework of problem domain within which features will be added.
37	Development by feature	It refers to a process where development is driven and tracked by decomposed list of small, client valued functions.
38	Individual class ownership	A process where the consistency, performance and conceptual integrity of each class is the responsibility of an assigned single person.
39	Feature team	A process to encourage doing design activities in small, dynamically formed teams as well as encouraging evaluating multiple design options before one is chosen.
40	Inspection	It refers to a process of defect-detection technique providing opportunities to propagate good practice, conventions, and development culture.
41	Regular builds	A process to ensure that there is always a demonstrable system available. It also helps to solve all synchronization issues as early in the process as possible.
42	Reporting results	A process of frequent and accurate progress reporting at all levels, inside and outside the project based on completed work.

2.2.2 The Agile map

Similar to the map of traceability shown before, this Agile map aims to provide readers with the quick overview of Agile. The Agile map could be illustrated:

AGILE

MAIN PRINCIPLES/PHILOSOPHIES

Welcome & respond to change quickly [78, 79, 80, 81, 77, 91, 92, 92, 104, 107, 108, 109, 111, 112, 113, 125]	Reducing waste [82, 83, 84, 89, 91]	Deliver working software frequently [81, 78, 77, 94, 104, 107, 108, 109, 112]	Promoting communication among team member [85, 87, 77, 93]	Simplicity [86, 88, 77, 93, 105, 106, 109, 114]	Cutomer collaboration [77, 104, 107, 108, 109, 111, 113, 114]	Satisfying customer [77, 104, 111, 114]	Motivating team members [77, 95, 114]

TOOLS TO ACHIEVE THESE PRINCIPLES

Extreme programming

Pair Programming [97, 98, 100, 101, 109, 126, 42]	On site customer [97, 100, 114, 118, 42]	Coding standards [97, 100]	System metaphors [97, 98, 100, 118]
Small release [97, 100, 118, 42]	Collective code ownership [97, 100, 118, 42]	Sustainable pace [97]	Planning game [100, 118]
Stand-up meeting [97, 100]	Refactoring [97, 98, 99, 100, 101, 110, 118]	Continuous integration [97, 98, 100, 101, 118, 42]	40 hour week [100, 118]

Scrum

Backlog [100, 102, 119, 1280, 42]	Common room [100, 128]	Sprint [100, 119, 128, 42]	Stand-up meeting [100, 119, 129]
Sprint planning [100]	Daily build [100, 119]	Sprint review [100, 119]	Scrum master firewall [100, 119]
Locks priority within Sprint [100]	Decision in 01 hour [100]	High level desigh[100]	Self-directed & organizing team [100, 119]

Lean sw development

Eliminate waste [100,102, 115, 116]	Built quality in [100]
Create knowledge [100]	Deliver as fast as possible [100, 116]
Decide as late as possible [100, 102, 116]	Amplify learning [102, 116]

Feature-Driven Development

Domain object model [100, 102, 117, 127]	Development by feature [100, 102, 117, 127]
Individual Class ownership [100, 127]	Feature Team [100, 127]
Inspections[100, 127]	Regular builds [100, 127]

2.3 Problems of traceability in Agile development process

Appleton [13] described a series of problems if we use traditional traceability approaches for Agile software projects because these practices have the following disadvantages: unnecessary creation of tracing artifacts and almost inevitable failure to accurately maintain these links; focusing on upfront activities and comprehensive documentation means that team members have to delay important tasks such writing codes and delivering executable products; or the creation of overheads to the change process itself, which actually makes change more difficult to be implemented.

In [22], Jane Cleland-Huang summarized feedback from agile project management group saying that even though the agreement was that the Agile manifesto should not be subject to change, most opinions strongly supported the core idea stating that traditional traceability approaches were generally detrimental to an Agile software project. It is suggested that traceability practices supporting Agile software actually need to be much leaner [22] because these practices need to be based on the depth knowledge of software developers and existing Agile software artifacts, e.g. test cases rather than existing traditional traceability approaches, which focus much on upfront activities and formal documents written before.

Current traceability methods seem to be specifically designed for traditional software development approaches. In [61], it was said that over the past decades, many traceability methods have been developed to support the traditional software approaches and these methods were designed based on the hypothesis that a formal requirement process was already in place. Arbi Gahzarian [61] stated that the wide industry adoption of Agile development methodologies in recent years has posed a particular challenge to the traditional traceability approaches.

Espinoza and Garbajosa [51] wrote that in an Agile practice, requirement traceability cannot be conducted as it is performed in traditional requirement management as required by ISO/IEC 12207:2008 or SWEBOK. The main explanation could be due to the reality that a formal system or user requirements specification document is normally omitted. This study [51] warned that if this problem is not properly handled, it could result in some serious challenges for other processes such as: change impact analysis, requirement change management or estimation. It is, therefore, strongly suggested that a new and specific traceability approach applied to Agile projects seems to be fundamental.

To cope with these problems or disadvantages of traditional traceability approaches, a number of researchers proposed solutions focusing on capturing specific traceability links. Christopher Lee et al [97] proposed the method of gathering requirements and maintaining traceability leveraging the Agile best practices. Ari Ghazarian [61] proposed a traceability pattern, in which traceability could be achieved through source code. Cao and Ramesh [38] suggested a strong traceability practice together with using explicit requirement negotiation, cooperative strategies for requirement engineering. Espinoza and Garbajosa [51] proposed the traceability model supposed to be less dependent on a specific development process.

Besides these proposals, some studies are said to cover a quite large spectrum of solutions suggested for achieving traceability in Agile projects. These studies will be summarized in the next chapter - Related Work, in which the outcomes of this thesis will be described to show what and how its outcomes could be supplemented or added to what other researchers have already done.

Chapter 3

Related Work

While formulating and coming up with the research topic, a preliminary search and review was conducted. This preliminary search result indicated that no systematic or traditional literature review discussing traceability in Agile software projects, had been conducted yet. When searching literature, however, a couple of studies were found to discuss issues related to traceability in Agile software development. Even though these studies belong to the same area of research, they do not focus on discussing issues addressed in this paper. Below is the summary of those studies:

In 2005, Spanoudakis and Zisman [143] conducted their study “Software Traceability: A Roadmap”. These two authors summarized and discussed most literature discussing software traceability up to that date. Particularly, this study first discussed types of traceability, to which the first research question of this paper mentions. Second, Spanoudakis and Zisman gathered and classified traceability tools into three categories: manual, semi-automated and automated. This literature provided descriptions of some traceability approaches. This study continued to summarize different approaches to deploy traceability activities. It could be deemed that this study was a systematic literature review summarizing previous literature discussing traceability issues. Thus, it is both good and comprehensive source of information related to software traceability. However, it seems that this study summarized traceability issues in traditional software, not Agile or Lean software development.

In 2007, Brad Appleton et al [13] did their study named “Lean Traceability: a smattering of strategies and solutions”. They gathered and summarized opinions from different roles in some organizations stating the negative aspects of doing traceability in Agile software projects. This study focused on describing traceability strategies as well as traceability practices that companies should take into consideration, for example: in order to have lightweight traceability, a company could use existing configuration management tools to capture tracing links. Some Agile practices, e.g Test-Driven Development, could help to obtain traceability. Results of this study could be referenced when identifying tools/approaches/methods to achieve lightweight traceability. However, this study is not the empirical literature, but summarized opinions from informal sources and then suggested strategies, practices without evaluation from both academia and industry sources.

In the year 2009, Marcus Jacobsson [79] conducted his master thesis “Implementing Traceability In Agile Software Development” to discover how traceability is handled in Agile software organizations. This study focused two main issues. The first issue is to describe tracing practices, which actually described the specific tracing links/relations. Marcus focused on describing why traceability links from requirements to other software artifacts are needed. The second is this study suggested general consideration, factors or conditions that should be taken into account before deciding to implement traceability. Specifically, Marcus talked about the size, the length and type of the projects. In addition, Marcus elaborated details about cost/benefits and which role in the team to implement a tracing link, for example: he explained what costs/benefits of implementing a link from a requirement to a test case. However, this study did not review any empirical study up to that time. Instead, this study was done mainly by interviewing some people, then describing feedback and then suggesting opinions. In addition, Marcus did not mention the research methodology stating how he extracted, summarized the data collected. However, this study is a good source to see details of specific tracing links and some factors needed to be considered before deciding the level of traceability for a system.

In 2012, Jane Cleland-Huang [23] with her study "Traceability in Agile Projects", summarized and synthesized previous literature, particularly two previous studies [13] and [79] suggested traceability levels such as: Basic, Retrieval. Basing on some factors such as: size, length and type of projects, Jan Cleland-Huang suggested which level of traceability could be matched with a specific project. This study specifically suggested two models of achieving traceability: Traceability Information Model (TIM) and Just-in-time traceability (JITT). The results of this study could be referenced to consider to which specific degree of traceability should be added to an Agile project and what tracing links need to be considered. However, this study did not mention what concrete tools/techniques are used to create, maintain and use tracing relations. Furthermore, this study neither collected feedbacks of practitioners nor reported findings of literature in a systematic way.

This thesis is conducted by reviewing existing empirical studies and collecting feedback & evaluations of practitioners to answer five main questions. First, results are intended to answer what are important types of traceability and what are reasons behind them. The thesis paper will also explore why some types of traceability considered as the important in traditional software processes, but *not* in Agile. Second, this thesis intends to collect feedback from practitioners identifying what mechanisms that are used to achieve traceability and evaluates if these mechanisms work *well* in the Agile environment. These answers could be deemed to supplement the related works mentioned above, particularly the comprehensive literature [32]. Furthermore, challenges and benefits of implementing traceability in Agile will also be reported from both industry and academia.

Taking a glance, this thesis attempts to be aligned with studies [23] and [79]. Jane-Cleland Huang in [143] suggested general models of traceability in several types and size of software projects, while Marcus Jacobsson [79] elaborated specific tracing relations, e.g, requirements to artifacts in particular. This paper will, however, take the different direction. First of all, two world maps of traceability and Agile with different and quite detailed nodes will be visualized to provide knowledge area for readers. Then, some specific nodes on two maps will be put together to formulate the research questions. After that, these nodes will be provided with results extracted from both the industry and academia findings.

Chapter 4

Research Methodology

To provide answers for research questions in this thesis, two main research methods were used: Industry interview and Literature review.

This chapter is organized as followed: a brief description of each research method is introduced. Then, the mapping of research questions is also presented in order to help readers gather information about: research questions, their aims and respective research methods in one table. Finally, the flow chart of the over all picture illustrates how research methods are connected and collaborated in this thesis with the aim of providing a research question with its relevant answer.

4.1 Brief description of research methods

4.1.1 Literature Review

AvChris Hart [132:13] defined Literature review research method as "The selection of available documents (both published and unpublished) on the topic, which contain information, ideas, data and evidence written from a particular standpoint to fulfill certain aims or express certain views on the nature of the topic and how it is to be investigated, and the effective evaluation of these documents in relation to the research being proposed". Colin Robson [136] considered the term 'literature' as what is already known, written down and relevant to our research projects. Reviewing literature is very common and traditional way that researchers use to gain knowledge areas when exploring researched problems. Levy and Ellis [94] defined "literature review" method as the process containing "sequential steps to collect, know, comprehend, apply, analyze, synthesize, and evaluate quality literature in order to provide a firm foundation to a topic and research method".

4.1.2 Snowball Sampling

According to WikipediA, Snowball sampling or change sampling or referral sampling is non-probability sampling technique where existing study subjects recruit future subjects from their acquaintances. Therefore, the sample group appears to grow like a rolling snowball. Vogt [155] stated that this technique is used for finding research subjects when researchers have difficulties in accessing hidden populations. Specifically, one subject gives researchers the name of another subject, which in turn provides the name of third subject and so on.

At the beginning of the thesis, searching literature related to the research problems provided low response and returned only a limited literature actually associated with the research questions. Thus, snowball sampling method was used to find additional important articles. Snowball sampling technique in this study mentioned to forward, backward and author search. On the one side, specifically, the thesis author used 'Backward' and 'Forward' search techniques. On the other side, the author contacted several researchers, who wrote specific literature or specially focused on the research area with the aim of finding more important studies. More details of this research method will be discussed in the next chapter.

4.1.3 Industry interview

The interview research method basically refers to the method, in which researchers ask questions and receive answers for those questions from participants or subjects. Hove and Anda [72] summarized that Qualitative research methods originate from sociology and anthropology, which were designed mostly by educational researchers and social scientists and the interview method is a frequently used technique for data collection within qualitative research.

Colin Robson [136] classified types or styles of interviews into three different categories detailed below:

1. Fully structured interview: has pre-determinded questions with fixed wording, usually in a pre-set order. The use of a greater number of open-response questions is the only essential difference from an interview-based survey questionnaire.
2. Semi-structured interview: the interviewer makes an interview guide that serves as a checklist of topics to be covered and a default wording and order for the questions. However, the order and wording are often substantially modified based on the flow of the interview and additional unplanned questions are asked to follow upon what the interviewee says.
3. Unstructured interview: the interviewer has a general area of interest and concern, but lets the conversation develop within this area. It can be completely informal.

The interview method has some major phases. First, the interview package including the interview guide and the list of questions, will be created. Next comes to the phase of conducting the interview, in which the author will focus on finding suitable interviewees and then implementing the interview. After that, data is transcribed, processed and analyzed. Finally, the audio data will be converted and exported to common format of text data for reporting results.

4.2 Mapping of research questions and research methodologies

Each research question, its aim and what research methodology needs to be used to obtain results answering for that question, are described in the Table 4.1:

Table 4.1: Research questions and its relevant methods

Research Question	**Aims**	**Methodology**
RQ1: What are the important types of traceability in Agile software projects?	To gather the list of traceability types and point out which types are important based on the results from the interviews conducted in industry (see more details in the section 7.2.1	Interview and literature).
RQ2: What mechanisms are used by Agile companies to achieve traceability?	To know mechanisms (tools, methods, frameworks, approaches, etc) that Agile companies are using to implement traceability	Interview.
RQ3: What mechanisms work well in Agile development organizations?	To gather opinions of practitioners evaluating if traceability mechanisms work well in Agile development organization	Interview.
RQ4: What challenges exist for realizing traceability in Agile organizations?	To gather opinions, feedback from the industry and literature identifying difficulties when conducting traceability in Agile organizations	Interview and Literature.
RQ5: What benefits exist in realizing traceability in Agile organizations?	To gather opinions, evaluations from the industry and literature identifying benefits when conducting traceability in Agile organizations	Interview and Literature.

Each research question and its corresponding research methodology can be visualized in the Figure 4.1:

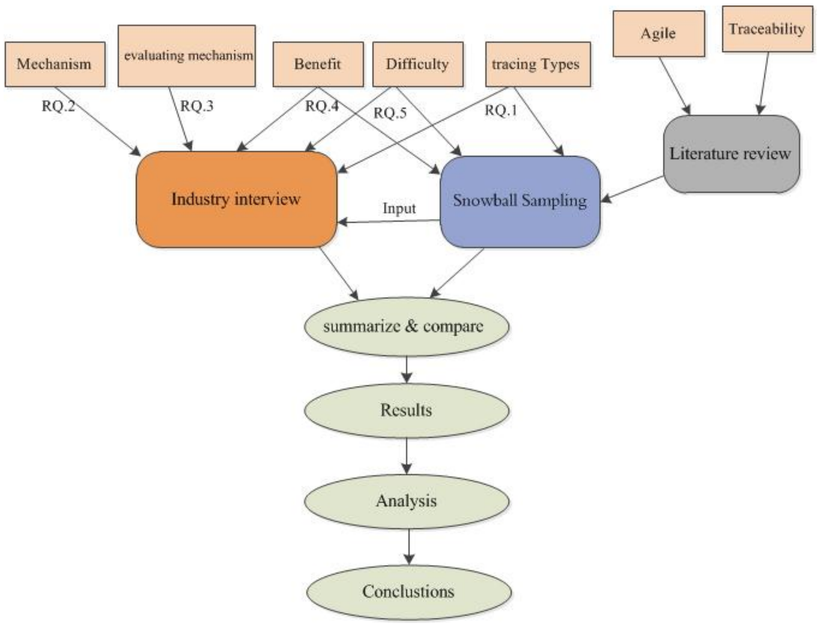


Figure 4.1: The flow chart of research methodology

Chapter 5

Conducting literature review

5.1 Motivation for the method selection

Before conducting a literature review, a researcher needs to understand the place of the review in research [157]. Therefore, two questions must be raised: What is the research about? Why a literature review method is needed for this research topic? Definitely, researchers understanding their own research topics could be able to answer the former question without difficulties. Answering the second question means that researchers need to identify the purposes of doing a literature review. According to Colin Robson [136], a literature review method has following purposes:

- Exploring main gaps in knowledge and identifying areas of dispute and uncertainty. Levy and Ellis [94] also supported this purpose when stating that a literature review helps researchers understand the existing body of knowledge by answering some questions i.e. what is already known? or what is needed to be known?
- Helping to identify general patterns to findings from multiple examples of research in the same area.
- Juxtaposesing studies with apparently conflicting findings to help explore explanations for discrepancies.
- Helping to define your terminology or identifying variations in definitions used by researchers or practitioners.
- Helping to identify appropriate research methodologies and instruments. Levy and Ellis [94] also mentioned this goal when writing that a literature review method could frame the valid research methodologies, approaches, goals, and research questions for the proposed study.

Before selecting specific research methods, the preliminary search was conducted to answer following questions:

1. Is there any systematic or traditional literature review already conducted for traceability in Agile software projects and to which extent? OR
2. Is there any empirical study evaluating traceability in Agile software projects?

To answer these questions, the author defined some initial searched terms and put them into 'Summon'-the search engine of Chalmers University of Technology or "Google Scholar". Presently, it is much convenient to start searching with "Google Scholar" since it can directly link researchers to specific sources or databases, where literature is stored. After reviewing returned hits, there was neither systematic nor traditional literature review related to the research topic of this paper. These initial searched strings were then directly searched inside popular electronic databases like: IEEE Xplore, ACM Digital Library, SpringerLink or Engineering Village because these databases are said to store most of quality literature in Software Engineering. After reviewing the 'title' and 'abstract' of literature, no systematic or normal review related to the research subject under investigation was found.

The literature review is a common method, to which degree almost every researcher needs to use in an attempt to discover existing knowledge of a research area. Webster & Watson [157:13] confirmed that "A review of prior, relevant literature is an essential feature of any academic project. An effective review creates a firm foundation for advancing knowledge. It facilitates theory development, closes areas where a plethora of research exists, and uncovers areas where a research is needed". In this thesis, therefore, the literature review is needed and used.

After the quick review of literature identified from the preliminary search, the author recognized that most articles discussed traceability issues in traditional software development methods, e.g. Waterfall, Linear models. Only a couple of articles are important and closely associated with the thesis topic. The thesis author contacted authors of these important papers and other specialists on traceability research [172] with the aim of finding additional important articles. Furthermore, tracking references, keywords, authors or citations of initial important papers was also conducted. These tips of searching literature could be considered as snowball sampling research method used for searching and identifying important articles.

5.2 Searching and Identifying literature

According to Levy and Ellis [94], conducting a literature review requires a researcher to walk through a process containing three stages: Input, Processing and Output. These three phases consisting of relevant sequential steps, are illustrated in the Figure 5.1:

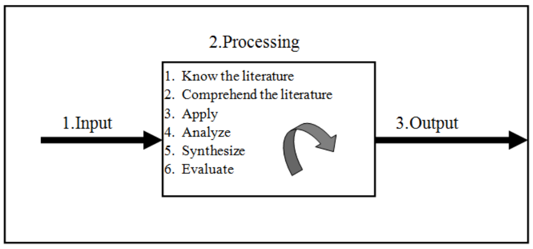


Figure 5.1: The three stages of effective literature review process

However, the review process in the Figure 5.1 was not strictly needed to apply to this thesis because of some reasons. First, the literature review is not the main method helping to provide answers to all research questions. Instead, it was used to provide the input for the main one- the industry interview. Second, as mentioned before, results of the preliminary search showed quite low hits, and then snowball sampling techniques was used. Therefore, the review process suggested by Levy and Ellis [94] was cut some steps. Below is the description of steps adapted to this thesis paper:

5.2.1 Validating quality literature

Davison et al [49] stated that using the peer-review process is essential because this process assures that researchers can “use published work with confidence, and use the works of others as stepping stones and cornerstones for advancing new concepts and insights”. However, the quality of all published materials is not equal. Thus, when looking for quality literature, we need to carefully see any work that is not peer-reviewed and we should limit or restrict use of these works. The quality from the world famous and top journals or articles will serve as the major base of literature review because this quality can provide sufficient theoretical background as well as leads for additional references on the specific subject matter.

Basing on the world top ranked list made for journals, Robert Feldt [59] selected and suggested the list shown in the Table 5.1 as the world top ranked journals in software engineering up-to-date:

Table 5.1: ISI Jouranls - Software Engineering

N.o	Journal	2011	2010	2009	2008	JSS
1	IEEE Transactions on Software Engineering (TSE)	1.98	2.22	3.75	3.57	Yes
2	Communications of the ACM (CACM)	1.92	2.35	2.35	2.65	
3	Empirical Software Engineering (ESEJ)	0.85	1.78	1.61	1.09	Yes
4	IEEE Software (SW)	1.51	1.51	2.04	2.10	Yes
5	IEEE Computer (Comp)	1.47	1.79	2.21	2.09	
6	ACM Transactions on Software Engineering and Methodology (TOSEM)	1.27	1.69	2.03	3.96	Yes
7	Information and Software Technology (IST)	1.25	1.51	1.82	1.20	Yes
8	Software and Systems Modeling (SoSyM)	1.06	1.27	1.53	N/A	
9	Requirements Engineering Journal (REJ)	0.97	0.86	0.93	1.63	
10	Software Testing Verification & Reliability (STVR)	0.96	0.76	1.63	1.05	
11	Automated Software Engineering (ASE)	0.86	0.81	1.27	N/A	
12	Journal of Systems and Software (JSS)	0.84	1.28	1.34	1.24	Yes
13	Software Maintenance and Evolution - Research & Practice (SMERP)	0.84	0.61	1.14	0.97	
14	IBM Journal of Research and Development (IBM JRD, IBM Systems Journal was merged into this one in 2009)	0.72	1.79	1.29	1.88	
15	Software Practice & Experience (SPE)	0.52	0.57	0.67	0.71	Yes
16	Software Quality Journal (SQJ)	0.42	0.75	0.98	0.95	
17	IET Software (IET SW, was called ‘IEE Proceedings - Software’ before 2007)	0.33	0.67	0.65	0.54	
18	Software Engineering and Knowledge Engineering (IJSEKE)	0.13	0.25	0.33	0.45	

5.2.2 Locations for quality literature

The searching literature process is to query academic literature databases such as: ScienceDirect, Wilson-Web, Elsevier, JSTORE, etc with the aim of gathering research materials associated with the research problem under investigation. Webster & Watson suggested that “the major contributions are likely to be in the leading journals. It makes sense, therefore, to start with the....[researchers] should also examine selected conference proceedings, especially those with a reputation for quality" [157:16].

Levy and Ellis [94] warned that inexperienced researchers make mistakes when using a keyword search approach and put it into only one or two literature database vendors. This method leads to two main limitations:

1. Very narrow literature background.
2. Shallow depth of literature background.

Normally, a vendor can only be able to provide a few journals and electronic resources. Obviously, using one or two literature databases can only obtain limited numbers of literature discussing the research problem. Webster and Watson noted that "a systematic search should ensure that you accumulate a relatively complete census of relevant literature" [157:16]. To cope with this limitation, the author of this paper used at least more than five vendors providing most of top ranked journals in the table 5.1 mentioned above. Specifically, main electronic databases are used such as: IEEE Xplore Digital Library, ACM Digital Library. The Table 5.3 is showing all main sources, from which literature is searched or queried:

Table 5.3: Locations for quality literature

Data sources	**Databases name**
Digital library	IEEE Xplore
	ACM Digital Library
	SpringerLink
	Engineering Village
	Science Direct
Search Engine	Google Scholar
	Summon (University search)

Apart from these main sources, a number of other electronic databases were used when using 'Summon'-the Library university search tool or "Google Scholar" because these search engines helped to directly link the researcher to access the databases of vendors, with whom Swedish Universities signed commercial contracts providing the legal access to these databases.

5.2.3 Keywords search

Keyword search refers to the activity of using a specific word or phrase to query quality literature databases in the attempt to find relevant literature. Researchers can put keywords into different search fields such as: title, abstract, author, etc. Levy and Ellis [94] stated that even though alternative keywords are used to search, some limitations may be hardly avoidable. Specifically, some keywords of literature in Software Engineering may have a limited life span. For example: in the period of 1980s-1990s, a lot of literature mentioned the phrase "requirement traceability" to actually refer to 'traceability' issues. Presently, "requirement traceability" could be used when referring exactly to the tracing direction from a requirement to other artifact or vice-versa. Another limitation of keywords is the use of technology specific terms or 'buzzwords' that appear and disappear from literature over time.

To cope with these limitations, Levy and Ellis [94] wrote that backward and forward search techniques can help researchers to go beyond keywords. 'Backward' and 'Forward' search will be covered in the next section. Below is the list showing the number of keywords used in this thesis:

1. Traceability

2. Software traceability
3. Requirement traceability
4. Type of traceability
5. Traceability link(s)
6. Tracing direction(s)
7. Purposes of traceability
8. Traceability goal(s)
9. Traceability benefits
10. Traceability challenges
11. Traceability tools
12. Traceability approaches
13. Agile philosophy(s)
14. Core values of Agile
15. Principle of Agile
16. Agile software development
17. Agile software development: review
18. Main Agile method(s)
19. 1 AND in Agile
20. 1 AND 16
21. 2 AND 16
22. (4 OR 5 or 6) AND 16
23. 9 AND 16
24. 10 AND 16
25. (11 OR 12) AND 16
26. (2 OR 3) AND (Agile OR XP OR Scrum OR Lean software OR Streamline)

5.2.4 Snowball sampling search

As mentioned before, Snowball Sampling method was used in this thesis as some tips of searching additional relevant articles. Specially, 'Backward' and 'Forward' search techniques were used. Webster and Watson [157] originally suggested the approach of backward and forward search for literature in Information Systems. According to these two researchers, backward search means that researchers review the citations for the articles identified after the keywords search process in order to determine articles which you should consider. Forward search, on the contrary, is to use the electronic version of the Social Sciences Citation Index to identify articles citing the key articles identified after searching literature with a specific keyword. Depending on this search approach, Levy and Ellis [94] elaborated and broke down backward and forward steps into some sub-steps:

1. Backward or Forward author search: these techniques refer to reviewing what the authors have published prior/following the article.
2. Backward reference search refers to reviewing the references of the articles obtained from keywords search noted before. On the contrary, forward reference search refers to reviewing additional articles that have been cited in the article.
3. Previously used keywords: this technique refers to reviewing the keywords noted in the articles obtained from keywords search mentioned above.

First, the author searched literature to draw traceability and Agile world maps, which are illustrated in Background chapter. Then, the author reviewed referenced literature in the maps in the aim to find important literature i.e. articles, books or journal. Important literature are ones discussing the research topic and literature, to which many other articles are referring. After that, a couple of important literature were searched with "Backward and Forward" search techniques to find more articles closely associated with the research problem. In addition, authors of important papers and some specialists focusing on traceability researches were contacted. For example: the world top researchers working at "Center of Excellence for Software Traceability" [172] were contacted to give comments for the traceability map and to find more papers related to the thesis topic. Or scholars writing the important papers in [13], were contacted to help the thesis author to search other researchers specializing in traceability studies.

5.2.5 When to stop searching

Levy and Ellis [94] stated that searching literature is an evolving process, in which a researcher is digging to find more and more studies related to the research topic. During the course of a research, searching literature is the continuous task. This searching process takes much time and seems to be a never-ending process forcing researchers, particularly the new ones, to ask themselves "At what point should the process of gathering additional relevant literature end?" [94:192]. Leedy and Ormrod [99] suggested that when reading a new literature piece, if one "will get the feeling that I have seen this (or something similar to it) before" [99:82], it may signal the searching process is nearly ending. The end of the search could be signaled when no new citations are discovered and articles cited in newly discovered literature have already been reviewed. Additionally, the end of the search could be indicated when observing that "You can gauge that your review is nearing completion when you are not finding new concepts in your article set" [157:16].

At the beginning of the thesis, the research topic was divided into specific sub-questions, each of which is provided with a relevant answer. Within each question, one or several keywords were picked and used for the searching process. Sub-questions could be seen as concepts or themes. The number of concepts or themes was fixed at the beginning. During the reviewed process, therefore, the thesis author was able to recognize where to spend more time for searching and when to stop.

5.3 Processing and synthesizing literature

5.3.1 Understanding the literature

Doing activities such as listing, describing and identifying can demonstrate the knowledge level of researchers, who need to read, review and extract meaningful information from identified literature [94]. For each identified literature, the thesis author firstly scanned its abstract, introduction or conclusion. If the literature is relevant, it will be stored. It depends on the importance of the literature to decide how many times it needs to be reviewed to extract useful information, for example: the article [143] is so important that the author could not remember how many times it had been reviewed. The thesis author used the “reading log” to make sure that each stored literature must be reviewed at least one time.

5.3.2 Comprehending literature

To comprehend the literature means to interpret, differentiate and summarize found papers. At the stage of a research, according to Levy & Ellis [94], not only can a researcher repeat information included in the article, he or she also needs to understand the meaning and significance of the information being reported. The author picked up each stored literature to review and extract significant information in order to prepare for next coming steps: summarizing and outputting results of the literature review. More details about comprehending literature activities will be described in chapter: Results and Analysis.

5.3.3 Applying literature

In this step, researchers mainly conduct activities such as classifying literature, relating them, or illustrating main concepts of reviewed papers. Levy and Ellis [94] considered that this step is actually concerned with two activities:

1. Identifying major concepts germane to the research questions.
2. Placing the citations in the correct categories.

To conduct these two activities, Webster and Watson [157] proposed two methods to structure and synthesize the reviewed literature. Webster & Watson stated that reviewed literature can be structured by concept-centered activities, in which concepts determine organizing the framework of a review. These two methods of structuring literature are illustrated in the sample 01

However, it is suggested that transferring author-centric to concept-centric seems to be easier and looks more structured. Levy and Ellis [94] adapted an idea made by Salipante et al [151] and suggested the use of this approach for synthesizing literature as shown in the sample 02. In case researchers want to add more dimensions to the concept matrix as mentioned in the sample 02 in order to handle the unit of analysis, this approach of synthesizing literature is suggested and shown in the sample 03:

Sample 01: Approaches to Literature Review	
Concept-centric	**Author-centric**
Concept X … [author A, author B, …] Concept Y … [author A, author C, …]	Author A … concept X, concept Y, … Author B … concept X, concept W, …

Sample 02: Concept Matrix

Articles	**Concepts**				
	A	B	C	D	…
1		✖	✖		✖
2	✖	✖			
….			✖	✖	

Sample 03: Concept Matrix Augmented with Units of Analysis

Articles	**Concepts**														
	A			B			C			D			…		
Unit of analysis	O	G	I	O	G	I	O	G	I	O	G	I	O	G	I
					✖				✖						✖
2	✖				✖	✖		✖							
….								✖	✖			✖			

The author selected the sample 02 to structure and synthesize literature reviewed because:

- It is the only the concept that needs to be centered and the author-centric is not necessary for extracting information of reviewed papers.
- Each concept or theme extracted from reviewed papers has the same meaning to an individual, a group or organization. Thus, there is no need to handle the unit of analysis.
- The results from reviewed papers are considered as the input to conduct the industry interview-the main research method.

Totally, 20 primary studies were carefully reviewed. They were synthesized according to Concept-centric as mentioned in the sample 02. The Table 5.4 shows each specific paper and the corresponding concepts described in the paper:

Table 5.4: Concepts Matrix for all primary papers

N.o	**Papers**	**tracing links**	**Type of link**	**Difficulty**	**Benefit**
1	Jane Cleland-Huang, 2012, "Traceability in Agile Projects", Computer Science, part4, pg: 265-275	X	X	X	X
2	Brad Appleton et al, "Lean Traceabilit: a smartter-ing of strategies and solutions", Configuration Management Journal 2007	X		X	
3	Jacobsson, M.: Implementing traceability in agile software development. Masters Thesis, Lund Institute of Technology (2009, January)	X	X	X	X
4	Cleland-Huang et al.: Best practices for automated traceability. IEEE Comp. 40(6), 2735 (2007). ISSN:0018-9162	X			
5	Lan Cao et al, 2009, A framework for adapting agile development methodologies, European Journal of Information Systems, pg: 332-343	X	X	X	X
6	Ratanotayanon et al, 2009, Supporting Program Comprehension in Agile with Links to User Stories, IEEE Computer Society-Agile Conference, pg: 26-32	X			X
7	C. Lee, L. Guadagno and X. Jia, "An Agile Approach to Capturing Requirements and Traceability", pp. 17-23, 2003.	X	X	X	X
8	Abdallah Qusef et al, 2010, "Recovering Traceability Links between Unit Tests and Classes Under Test: An Improved Method, Software Maintenance (ICSM)", IEEE International Conference, pg: 1-10	X	X	X	X
9	Cao and Ramesh, 2008, "Agile Requirements Engineering Practices: An Empirical Study", IEEE Computer Society, vol.25, issue.01, pg: 60-67	X	X	X	X
10	Espinoza and Garbajosa, 2011, A study to support agile methods more effectively through traceability, Inovations in Systems and Software Engineering, Vol.7, No.1, pg:53-69	X	X		X
11	Lorena and Gotel, 2007, Story-Wall: A Concept for Lightweight Requirements Management, 15th IEEE International Requirement Engineering Conference, pg:377-378	X	X		X

12	Vassilka Kirova et al, 2008, “Effective Requirements Traceability: Models, Tools, and Practices”, Whiley Online Library, Vol.12, Issue.4, pg:143-157	X	X	X	X
13	Ghanam and Maurer, Linking Feature Models to Code Artifacts Using Executable Acceptance Tests, Computer Science, SpringerLink, Vol.6284/2010, pg:211-225	X	X	X	X
14	Larks and Tobias, 2009, “Towards End-to-End Traceability”, IEEE Computer Society, Software Engineering Advances, pg: 465-470	X	X	X	X
15	Arbi Ghazarian, 2008, ”Traceability Patterns: An Aproach to Requirement-Component Traceabilit in Agile Software Development”, Proceedings of the 8th WSEAS International Conference, Applied Computer Science	X		X	
16	Hayes et al, 2009, “Towards Traceable Test-Driven Development”, IEEE workshop, pg:26-30	X		X	X
17	Jyothi and Rao, 2011, “Effective Implementation of Agile Practices”, International Journal of Advanced Computer Science and Applications, Vol.2, No3	X		X	X
18	Lucia and Qusef, 2010, “Requirements Engineering in Agile Software Development”, Journal of Emerging Technologies in Web Intelligence, Vol.2, No.3	X			X
19	Paetsch et al., 2003,Requirements engineering and agile software developement, WET ICE 2003, Proceedings. Twelfth IEEE, p.308-313	X		X	
20	Martin et al., Barriers to Adopting Agile Practices When Developing Medical Device Software, Springer-Erlag Berlin Heidelberg 2012, p.141-147	X		X	

Chapter 6

Conducting industry interviews

6.1 Motivation for the Interview method

Typically, we need to conduct in-depth interviews when seeking information about individual or personal experiences from people about a specific topic [71]. The interview, in this study, is needed in order to gain insight into certain issues related to traceability in Agile software project. Specifically, the interview method could help to gain information:

- How practitioners decide and which degree of traceability should be implemented in an Agile project.
- The beliefs and perceptions of interviewees about issues related traceability in Agile.
- The knowledge, experiences, feelings or emotions of interviewees when answering a series of questions.
- To see how internal factors within an organization/company influence issues-related traceability.

After getting and basing on the input from the literature review, the industry interview method will be used as the main driver to provide answers to all of five research questions in this paper.

6.2 Interview Preparation Plan

6.2.1 Finding interviewees

Appropriate selection of interview subjects is essential in getting precise and useful information. People, who were directly involved in software requirement management, particularly in software traceability were selected as the right candidates for interviews. Some roles in a project like Project manager, Software Configuration manager, Requirement manager, experienced developers/testers were considered in this regard to make the research study more authentic and reliable. The subjects selected for interviews were most in the same classification.

Totally, 14 interviewees of 12 companies (02 companies with 02 interviewees each) were interviewed. 12 interviewees have 15-20 years working experience in software development unit. These people are holding important positions. 01 developer and 01 tester have less than 05 years of experience. Before searching for suitable interviewees, the formal letter was sent to a gatekeeper in each company. Most gatekeepers are project managers, so that they helped to find the right interviewees, who should have solid knowledge and experience related to this research topic.

6.2.2 Developing the interview guide

The interview guide is created to simply guide the interview, which is not the questionnaire. According to Hennink et al [71], the interview guide is neither called a questionnaire nor used in the same way as a questionnaire. The questionnaire is a structured research instrument for quantitative research, with predominantly closed questions that people are asked to respond to. In the in-depth interview, however, interviewees do not respond as the way they do with a questionnaire. They actually participate in an interview and tell their own story. The interview guide of this study was structured with sequential sections described below.

Structuring the interview guide

In this phrase, some tasks were created and specifically described as followed:

1. *Creating the introduction:*
   Some introductory points were included in this guide with the aim of the interviewer (thesis author) to start an interview, communicating with interviewees. Typically, the interviewer introduced himself, explaining what was the main purpose of the study, what would be done with data and information collected from interviewees. The interviewer also mapped out the outcomes of the research and mentioned some ethical issues, for example, how some confidential information about interviewees and their organizations are protected. Also, the permission from interviewees was obtained to record their voice.

2. *Creating opening or warm-up questions:*
   After introducing himself, the interviewer should continue with some general and opening questions. Hennink et al [71] stated that the goal of opening questions is to continue building rapport with the interviewees so that participants could get comfortable feelings to tell their truly stories when being once the interviewer comes to important questions mentioned in the next section. Warm-up questions can be related to the background of interviewees. In this study, specifically, questions about their companies in general were asked with the aim of supporting key questions mentioned below.

3. *Key or important questions:*
   Key questions is the central part of the interview guide. As guided in [71], important questions are essential ones, which are designed to provide important information or data answering research questions. Key questions are placed at the central part of the interview guide to allow time for rapport to be set up between both sides: the interviewer and interviewee. Rapport is very crucial because it helps an interviewee to feel comfortable to share their real stories, experience related to the study. When asking key questions, the interviewer could add up some probe questions to gain more detailed data or information.

   During interviews of this thesis, basing on a specific research question, one or several key questions were extracted, designed. Specifically, after reviewing literature, some themes/concepts/points were filtered and extracted. These concepts were used to formulate important questions related to traceability in Agile software projects. In the interview guide, each key question was at least followed by one probing question.

4. *Closing questions:*
   Like any story, it starts with opening, then going into details and it will be finally wrapped up. The guide needs to include some ending questions. Hennink et al [71] strongly advises that even though important information has been collected, it is not good practice to simply end the interview without asking some closing questions. The interviewer needs to slowly reduce the rapport and create a distance again before leaving the interviewee. Similarly to warm-up questions, ending questions could be very general. In this thesis, for example: ending questions were raised to interviewees to answer if formal trainings about traceability should be provided to new employees in the future.

5. *Pilot-testing:*
   According to Hennink et al [71], the interview guide needs to be pilot-tested because it is difficult to foresee how interview subjects will interpret the questions included in the guide. Typically, it is suggested that pilot-testing should be conducted with people, who share the same characteristics as the actual interviewees. In this thesis, the interview guide was firstly drafted and it was evaluated several times by the supervisor of this thesis. Then, the author of this guide tried to answer as if he was an interviewee. Next, a post-doctor specializing in software technology research at Chalmers University of Technology was asked to answer interview questions. After this researcher finished, some questions were raised with him as followed:

   - Did you understand the questions right away?
   - Were concepts, sentences and words familiar?
   - Need some questions be re-phrased?
   - Was the order of questions logical and reasonable to you?
   - Was the interview guide too long to answer in about 1.5 hour?

   After the pilot-testing, this guide was revised and then evaluated again several times by the thesis supervisor.

6.3 Conducting interviews

During this phase, the interviewer needs to conduct multiple tasks mentioned below:

1. Getting acquainted with interviewees with a few words.
2. Establishing rapport and creating a comfortable atmosphere.
3. Listening and responding to the interviewees by asking probings or follow-up questions.
4. Motivating interviewees to go into details of their stories.
5. Being sincerely interested in listening to stories of interviewees, eve information may not be related to the research topic.

The author of this paper mainly focused on two sequential subsections described below:

Establishing rapport

Totally, 14 interviewees from 12 companies participated in the interviews, including 07 face-to-face interviewees and 05 telephone interviews. When meeting with each subject, a small talk was started to bring comfortable feelings for the interviewee. 06 face-to-face meetings and 04 telephone meetings were conducted at the office of interviewees, 02 interviews were conducted at the house of interviewees. It could, therefore, be confirmed that interviewees felt safe and comfortable.

6.3.1 Asking and motivational probing

Before each interview started, a note was put in front of the author as a guideline: focusing on listening to the interviewees or accepting even irrelevant information because the key point to elicit the story of the subject. However, the author also tried to be the interacting interviewer when needed. For example: the author responded and explained questions related to definitions of some Agile practices, or responded to subjects with questions "Could you please tell me more?". These questions were very helpful to motivate interviewees to elaborate their story.

It also depended on situations to add up or to remove some questions, even though the author tried to follow carefully prepared interview guide. For instance: the interviewee of company B talked a lot and very fast. He directed the author at the beginning and avoided some key questions or provided irrelevant information. However, by using some types of motivational probes as mentioned in [71] such as: 'OK' or 'ah-ha', this interviewee was guided in the right direction.

6.4 Processing and Extracting interview data

To process and extract data from audio tapes, a software named 'NVIVO' was used. NVIVO is the most common and popular software suggested for coding, storing and transcribing from audio data to text data. This software is legally provided by Gothenburg University for study purposes.

6.4.1 Processing data

In this thesis, the Qualitative research was followed. Hennink et al [71] wrote that Qualitative research is a broad term and allows researchers to examine people's experiences in details. Specifically, experiences and stories of people were captured by the in-depth industry interview. The main reason for this selection is that Qualitative research allows the thesis author to discover issues from perspectives of practitioners working in Software companies, to understand the meanings and interpretations that interviewees give their behaviors, opinions. For instance: interviewees were asked to tell their experience of using traceability software tools in their companies.

Sequential steps below will describe how data is coded, transcribed, stored and updated:

1. Coding and transcribing data:
   According to Colin Robson [136], there are several approaches to qualitative analysis such as: Thematic coding , Grounded theory and Quasi-statistical. However, Thematic coding approach was selected to code data. This coding approach is generic to the analysis of qualitative data. This coding approach helped to report feedback, experience, meanings and real stories of experts telling about how traceability is implemented in Agile projects. To follow this approach, the thesis author conducted the following activities:

   - For each audio interview tape, the author listened the first time in order to be familiar with data recored.

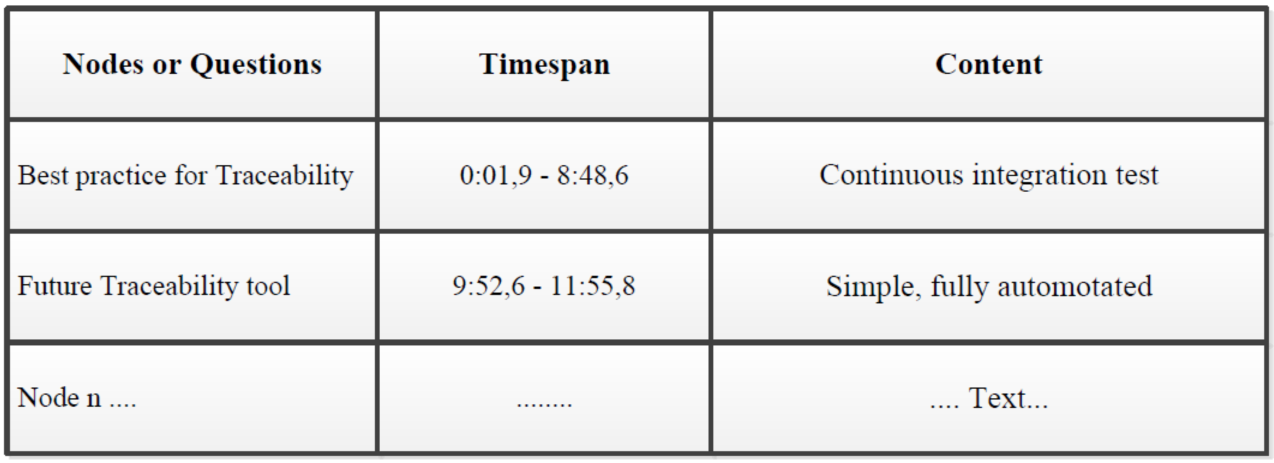
- Defining themes, which can be key questions in the interview guide or main concepts extracted from reviewing primary studies.
- Almost all data were coded and labeled.
- The author tried to write down exactly what interviewees actually said in audio files. This can be done quite easily with NVIVO, which provides different paces of playing audio file. Thus, the researcher could listen and capture almost exactly what is being said in the audio.
- After writing on the Interface of NVIVO, these texts were dragged and dropped to relevant nodes, which were defined and labeled before.
- Nodes containing text could be grouped under a theme. For example: a theme named "traceability mechanism" consists of 03 different nodes, which were labeled as: 'USED', 'future' and 'NOT-used'.

After writing down text for each node, this text then was saved. The process continued until the end of the audio tape. Finally, the full transcript for each interviewed audio was done.

2. Storing and updating data:
NVIVO software tool allows researchers to store a lot data in its database set up on a computer. After around 15minutes transcribing, transcribed data is automatically notified to be saved in case the user did not press 'save' button on the Interface of NVIVO. Thus, it can be said that it is very safe to work with this tool. Within the GUI of NVIVO, it is quite easy to go back to modify nodes, themes or transcribed text whenever the user wants.

6.4.2 Extracting and Gathering data

The data stored in the database of NVIVO can be directly converted to different data formats, particularly very common formats such as: Microsoft Word, Acrobat Reader pdf. The extracted data is in the table format. Typically, the table has three main columns. The first column shows all 'Nodes' that are coded in previous steps and as mentioned before. Nodes names were labeled the same as the interviewed questions in this thesis. The second column is to report the 'Timespan' that corresponds to each node. The last column is 'Content' that contains what the researchers have written down during the transcribing step. How many rows in the table totally depends on how many nodes created during coding and transcribing steps. That means the number of table rows is equal to the number of created nodes. The Figure 6.1 illustrates the format of data extracted from NVIVO database:



Nodes or Questions	Timespan	Content
Best practice for Traceability	0:01,9 - 8:48,6	Continuous integration test
Future Traceability tool	9:52,6 - 11:55,8	Simple, fully automotated
Node n	....	.... Text...

Figure 6.1: The format of data after transcribing

Chapter 7

Results and Analysis of this thesis

7.1 General information of companies and primary studies

7.1.1 Characteristics of reviewed literature

Publication year of primary studies

Referring to section 5.3.3, a total of 20 literature were selected as primary studies. This section shows some general characteristics of primary studies:

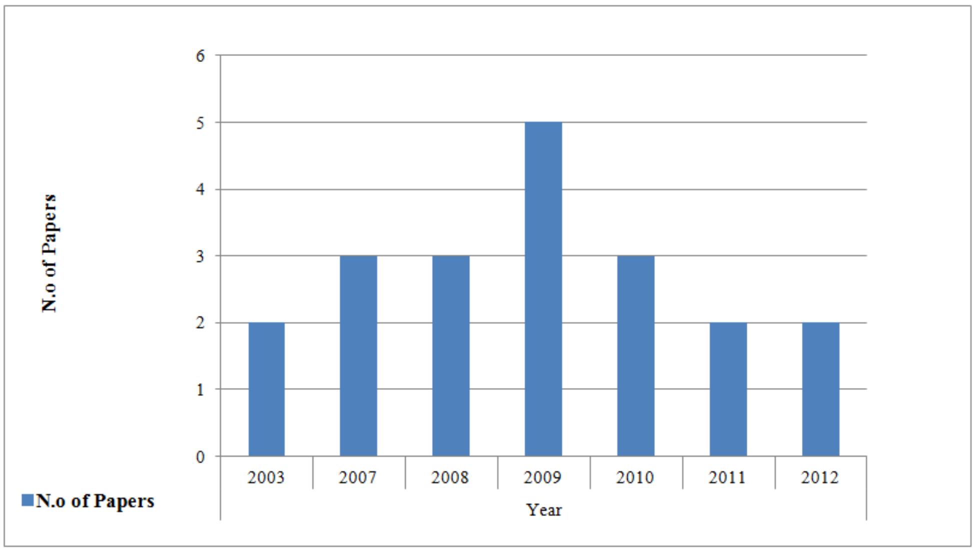


Figure 7.1: Primary studies by public year

Looking at publication years of primary literature, it is recognized that literature were published from 2003 to date. This result could be understandable since Agile software development was first introduced in 2001 and seems to have actually been growing over the past few years. The Figure 7.1 shows that the number of primary studies peaked in the year 2009 and dropped in 2010. This tendency may probably mean that after the year 2009, the topic of this thesis may have gradually drawn less interest of researchers. However, the difference of 02 papers between 2009 and 2010 is too small. Thus, more years is needed in order to track and reflect the right reason for this tendency.

Context of studies

Primary studies were classified into two categories: Industry and Non-Industry. The former only refers to the industry reports. The latter covers studies conducted by different research methods such as: case study, survey, action research, experiment, conceptual analysis, etc. The Figure 7.2 demonstrates literature according these two categories:

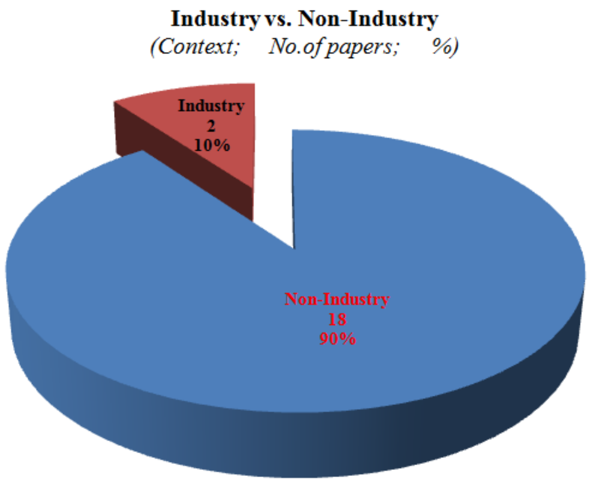


Figure 7.2: The Industry vs. Non-Industry distribution of literature

The Figure 7.2 shows that only two papers were conducted in the Industry context. This could be deemed that practitioners were paying less interest in this research topic or the topic was new in the industry. On the contrary, 90% of the primary studies were conducted in the Non-Industry context. This could be understandable since it seems that academic researches probably need to be conducted *before* feedback or reports conducted in the industry for any new research area.

7.1.2 Interviewed companies

Company overview

Totally, 14 interviewees representing twelve companies were interviewed (02 companies with 02 interviewees). Some general information about these companies is shown in the Table 7.1:

Table 7.1: Descriptions about interviewed companies

Company	Description	Main product
A	A global company with head office in Sweden. It was established in 1876 and has 109,214 employees working in almost every country on the globe	Telecoms network service and products.
B	It is a global company with head office in California, United States. It has more than 43.000 employees	It sells chips, tools and semi-conductor products in a wide variety of products ranging from a cell phone to an airplane.
C	A global IT company with its head office in Sweden. It is a member of the global group with more than 100,000 employees and the company has 5,500 employees	Around 80 % IT products support for different products made by other members within the group.
D	A small software company located in Gothenburg-Sweden and has around 40 employees	It makes software products in two main areas: Telecoms and Car industries.
E	An IT consultant company with head office in Stockholm-Sweden. The company is operating mainly in Nordic region and has over than 1,700 specialists	Its products cover different solutions for software product systems.
F	A small IT company with over 30 software developers located in Gothenburg-Sweden	It mainly makes web applications in E-commerce and banking sectors.
G	An IT consultant Swedish company operating in the Scandinavien area with more than 100 specialists	It mainly covers software solutions in Telecoms domain.
H	A global corporation with its head quarter in Sweden. It has over 13,000 employees working around the globe.	Products are in main areas such as: Military industry, Aviation, Navigation.
I	A Swedish IT consultant company with one office in Gothenburg-Sweden and the other in China and has around 60 employees for each site.	It mainly covers solutions for software systems in Telecoms.
J	It is the Sweden country office of the global IT and business group having more than 72,000 professionals and operating in more than 40 countries in four main continents.	This Swedish country office mainly provides IT solutions in Electricity and Water sectors.
K	A software vendor with more than 50 software developers located in Stockholm-Sweden.	It mainly works with partners in the Telecoms industry.
M	The leading consultant group is headquartered in France, with over 120,000 employees working in more than 40 countries	This Swedish country office provides IT products and services for main clients such as: IKEA, Sony, Ericsson, Energy sector.

Interviewed companies have been classified into 03 groups. The first is consultant companies that mainly focus on IT business consulting activities. Next comes to global corporations that have at least more than 50.000 employees and operate through out the global. The last is medium and small companies, which have less than 50 employees and mainly operate within the Swedish market. The Table 7.2 lists down companies according to these three groups:

Table 7.2: Types of interviewed companies

Consultant company	Global company	Small/medium company
E	A	D
G	B	F
I	C	K
J	H	
M		

As this topic focuses on discovering how Agile software companies implement traceability activities. It is, therefore, necessary to see and evaluate companies from two perspectives: 'Agile' and 'traceability'. These two perspectives are illustrated sequentially in the following sub-sections:

Understanding purposes of traceability

Before giving main questions, each interviewee was asked if he or she knew why conducting traceability is needed regardless of software development methods used. In other words, interviewees were asked if they know about purposes of achieving traceability in a software project. Each interviewee selected one of the five choices as described as followed:

- 1 = Don't know it at all. Thus, never do it
- 2 = Heard about it and did similar tasks in a couple of times
- 3 = Know about it, do it sometimes
- 4 = It is an integral part of the development process. Thus, do it quite often
- 5 = It is extremely important. Thus, always do it

There are 10 main purposes as covered in the section 2.1.3 and the Figure 7.3 describes the experience, insight of interviewees about traceability in a software project:

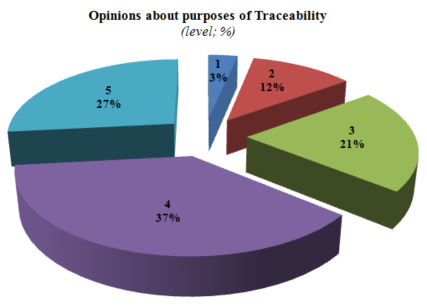


Figure 7.3: Interviewees answered about purposes of conducting traceability

The Figure 7.3 shows that only 3% of the total number of answers that interviewees did not know about purposes of achieving traceability. It means that almost all interviewees know and understand the importance or purposes of traceability. However, only 27% of total answers considered traceability as the very important task that needs to be conducted. It probably means that implementing traceability may face many challenges and consume much resource for companies while it could bring in few benefits. These issues will be discussed in details in RQ4 and RQ5.

The level of Agile used in companies

In this paper, four level of use was defined as followed:

- 1 = Not know about it. Thus, don't do it
- 2 = Know about it, but only do it sometimes
- 3 = It is a part of normal process, thus do it most of the time
- 4 = Central and always do it

42 Agile practices were defined as in section 2.2.2 of the Background chapter. Each interviewee selected 01 level for each practice. The Figure 7.4 is showing level of Agile practices that companies are using:

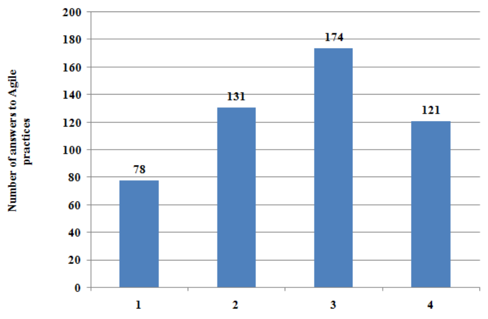


Figure 7.4: The level of use in Agile companies

Specifically, in the Figure 7.4, levels 1, 2, 3 and 4 defined above are showing actual degree of applying Agile methods in interviewed companies. 12 companies gave the total of 504 answers for 42 Agile practices.

The percentage corresponding to each level in the Figure 7.5, e.g. at level 1, means that companies did not know 15 % of total 42 practices defined in this paper. It could be deemed that these Agile practices may not be popular in the industry,especially for small and medium companies . In addition, only 24% of total answers stated that these Agile practices were always implemented in their companies. It could probably mean that companies may be using a combination of different development methods rather than only Agile practices in order to meet their needs.

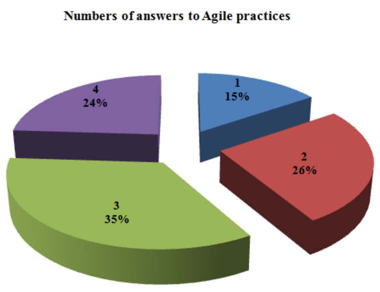


Figure 7.5: The level of use and its percentage in Agile companies

Companies evaluated the effectiveness of Agile

For this evaluated criteria, four scales were defined as followed:

- 1 = Not effective at all
- 2 = Relevant, but not helpful
- 3 = Effective
- 4 = Very effective

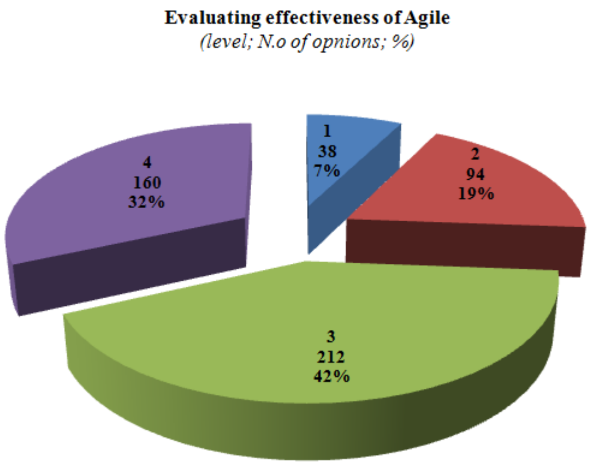


Figure 7.6: Companies evaluated the effectiveness of Agile practices

The Figure 7.6 shows that about 32% of total answers evaluating that Agile practices are very effective. It could imply that companies considered 68% of total Agile practices may not be as very effective as other development practices. However, it is very difficult to have the common criteria for companies to evaluate the effectiveness of Agile practices since evaluating Agile practices depends on many factors, situations such as: project, knowledge & experience, resources or regulations of companies.

Understanding about the principles of Agile

To evaluate the understanding level of each interviewee about principles of Agile as mentioned in the section 2.2.1, five scales were defined as follows:

- 1 = Strongly disagree
- 2 = Disagree
- 3 = Neither agree nor disagree (Not know)
- 4 = Agree
- 5 = Strongly agree

The interviewees filled in answers to illustrate their understanding main principles of Agile practices. The

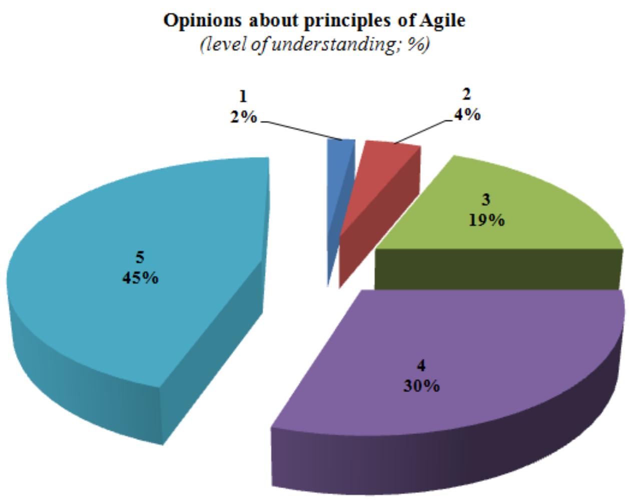


Figure 7.7: The level of understanding about Principles of Agile

Figure 7.7 shows how interviewees understand main principles of Agile. For example: at the level 3, interviewees said that they have no ideas about 19% of total number of Agile principles. It could be explained that interviewees may not be familiar with the words/phrases/terms defining these principles, though they could probably understand the purposes or the essence of these principles.

Checking the evaluation of companies

In this section, 04 data tables show data recorded for 04 respective evaluated factors: the levels of Agile and traceability, the evaluation for Agile principles and effectiveness. In four sequential sub-sections below, the data table and ANOVA test results will be provided. The purpose of using ANOVA test is to see if there is a significant difference between 03 groups of companies in understanding, evaluating or commenting some aspects of 'traceability' and 'Agile'. The result of this test will not draw any conclusions for this thesis, but could be used as the reference to reflect on the knowledge & experience of interviewees, the level of Agile used and the understanding of 'traceability' and 'Agile' in each company.

1. *Understandings about traceability*:
   When asking if traceability is extremely important that companies always need to implement, interviewees replied with numbers in the Table 7.3:

   Table 7.3: Understanding the importance of traceability

Consultant company	**Global company**	**Small company**
4	2	3
2	2	0
10	0	5
2	0	
2		

   To evaluate traceability as the extremely important, a company will select 10 purposes of doing traceability in the section 2 of HANDOUT-02 (see in the Appendix). For example, in Table 7.3 one consultant company selected all 10 purposes while each of two global companies selected only 2 purposes.

   Firstly, we will calculate the following values for each group:
   $\sum x$: is a sum of all the marked choices of that group.
   $\overline{X}$: the mean of that group's choices
   $\sum(x^2)$: a sum of the square of the group's choices

   Next, we will calculate these values for the entire set of cases:
   $\sum(\sum x)$: summing the three values for $\sum x$ that we computed previously
   $\sum[\sum(x^2)]$: summing the three values for $\sum(x^2)$ that we computed previously

   In this case, we shall get these values for consultant companies:

   $$\sum x = 4 + 2 + 10 + 2 + 2 = 20$$

   $$\sum(x^2) = 4^2 + 2^2 + 10^2 + 2^2 + 2^2 = 128$$

Doing the same computation for the other two groups will give us the following values:

Table 7.4: Three first basic values for each group

Stat.	Consultant company	Global company	Small company
$\sum x$	20	4	8
$\overline{X}$	4	1	2.7
$\sum(x^2)$	128	8	34

Then,

$$\sum\left(\sum x\right) = 20 + 4 + 8 = 32$$

$\sum\left(\sum(x^2)\right) = 128 + 8 + 34 = 170$

Now, we need to calculate three different sum of squares values: the sum of squares total, the sum of squares between, and the sum of squares within. Then, we will compute the mean squares for between groups and within groups. Finally, we will compute the F statistic by dividing the mean squares between by the mean squares within. So to begin,

$$\text{SS total} = \sum\left(\sum(x^2)\right) - \frac{\left[\sum\left(\sum x\right)\right]^2}{N} = 84.67$$

$$\text{SS between} = \sum \frac{(\sum x)^2}{n} - \frac{\left[\sum\left(\sum x\right)\right]^2}{N} = 20$$

SS within = SS total - SS between = 84.67 - 20 = 64.67

Next,

$$\text{MS between} = \frac{SSbetween}{dfbetween} = \frac{20}{n(groups)-1} = \frac{20}{3-1} = 10$$

$$\text{MS within} = \text{MS between} = \frac{\text{SSwithin}}{\text{dfwithin}} = \frac{64.67}{\text{N}-\text{n(groups)}} = \frac{64.67}{12-3} = 7.19$$

Finally,

$$\text{F} = \frac{MSbetween}{MSwithin} = \frac{10}{7.19} = 1.39$$

2. *The level of Agile used in companies:*

When asking for opinions about the level of Agile that companies are applying, interviewees replied with the figures in Table 7.5. In this thesis, 42 practices of Agile were evaluated. The data in Table 7.5 reflects the

Table 7.5: The level of Agile used in companies

Consultant company	**Global corporation**	**Local company**
8	1	20
15	10	4
20	8	3
5	11	
16		

evaluations for the level 3 of HANDOUT-01. For example, one global company selected only 01 practice at the level 3 while one consultant company selected 20 practices at this level.

Following the formula to compute F detailed above, we got the value: F = 0,77

3. *Companies evaluated the effectiveness of Agile:*
   When asked to give opinions about the effectiveness of Agile, interviewees replied with the figures in Table 7.6:

Table 7.6: Opinions about the effectiveness of Agile

Consultant company	Global company	Small company
3	11	16
13	9	6
18	18	20
18	9	
19		

Interviewees evaluated this criteria in the HANDOUT-01. In Table 7.6, companies selected the level 4 to evaluate the effectiveness of 42 practices. For instance: each of two consultant companies selected 18 practices at the level 4 while each of two global companies selected the same number 9 practices at this level.

Following the formula to compute F, we got the value: F = 0,20

4. *Companies evaluated main philosophies of Agile:*
   When asked to give opinions about main philosophies of Agile, interviewees responded and the results are in Table 7.7:

Table 7.7: Understanding main philosophies of Agile

Consultant company	Global company	Small company
7	5	7
5	5	0
0	2	5
7	0	
0		

Companies expressed their understanding level by selecting 08 main philosophies of Agile in the section 1 of HANDOUT-02. For example, two consultant companies did not select any philosophy (0 value) while two global companies selected 05 philosophies (5 value).

Following the formula to compute F, we got the value: F = 0,10

We got 04 different values of F, corresponding to 04 filled in questions that interviewees filled. Now, we need to consult an F table containing critical F values to see whether our results are significant or not. In this case, we had 2 degrees of freedom in the numerator (MS between) and 9 degree of freedom in the denominator (MS within). Looking at an F table, this would give us a critical F value of approximately 4.2565 at the .05 probability level. As we can see, our results were *not* significant at the .05 probability level because four calculated F values are 1.39, 0.77, 0.20 and 0.10 were *smaller* than the critical F value for the .05 probability level, 4.2565.

We could say the following: There is *not* a significant difference among 03 groups of companies in understanding and evaluating the importance of ‘traceability’ and ‘Agile’. In $\Big(F(2,9) = (1.39, 0.77, 0.20, 0.10), p > .05\Big)$, the first value for our degrees of freedom, 2, is equal the number of groups minus one. The second value, 9, is equal the total sample size or number respondents (interviewees), 12, minus the number of groups, 3. Four calculated values of F were very small in comparison to the critical F value. It could be mainly explained that the data was too small (each company group has only 3 or 5 members). It could imply that if many more companies were interviewed in this thesis, the calculated F values would be bigger, showing that there may be significant difference among groups. As mentioned before, however, the statistic test did not draw any conclusion, but was used to see if the representatives (the interviewees) of 12 companies could have equal or same basic knowledge and experience of understanding ‘traceability’ and ‘Agile’ before they started answering main questions.

7.2 Research Questions

7.2.1 RQ1: What types of traceability are important for software development in organizations subscribing to being Agile?

Aim: tracing links could be classified and named with different types. This question is to discover what types of tracing links are important. It also aims to see why some other types are *not* important.

1. **Results from reviewing papers:**
   During the review process, types named such as: *rationale or evolution or satisfy*, were only found in few papers while tracing links/relations were very often mentioned in most of 20 primary studies. For example: it is very often to find tracing links/relations e.g. from a requirement to a test case or from a code class back to an user story in literature. Thus, it is necessary to include the "tracing links" concept into the Concept Matrix-the table mentioned in section 5.3.3 in order to avoid missing important information.

   As elaborated in section 2.1.7 of the Background chapter, there are eight types of tracing links or relations being discussed in this thesis. Figure 7.8 illustrates: each type of traceability, its respective number of papers and the percentage:

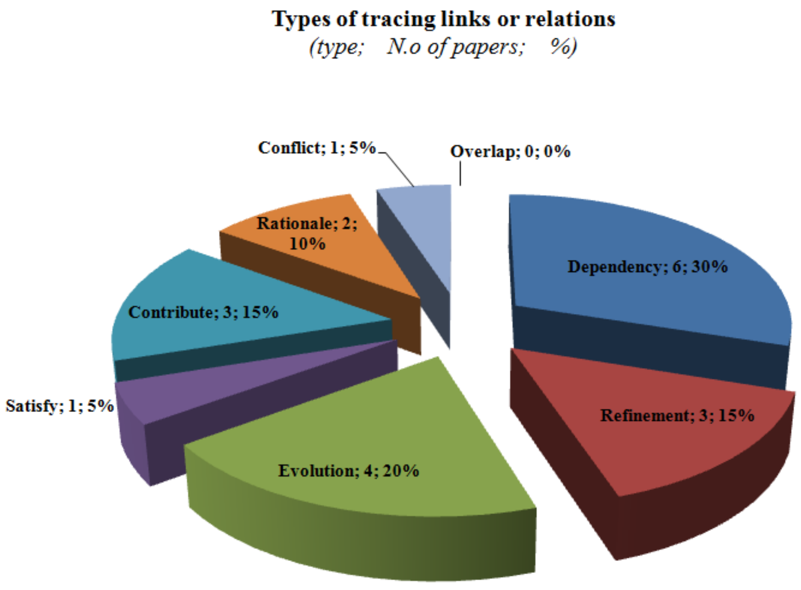


Figure 7.8: The types of tracing links or relations distributed by literature

The Figure 7.8 states that ‘Dependency’ is the most important type as 30% of papers stated. The second most important is ‘Evolution’, which was mentioned by 20% of reviewed papers. Both ‘Refinement’ and ‘Contribute’ were equally important and accounted for 15%. ‘Satisfy’ and ‘Conflict’ were only mentioned in 01 paper for each type, even no paper discussed the ‘Overlap’. Detailed description for each type of tracing relation is elaborated in the Table 7.8:

Table 7.8: Findings extracted from primary studies

No.	Type of traceability	Description
1	Dependency	This type could represent for different links from a requirement to other artifacts as stated in papers [71], [79] and [89]. Dependency in paper [67] symbolized links tracing from a test case back to a code element or an user story. In the paper [67], this type was used to illustrate the link between a feature model (design) to a source code class.
2	Evolution	Papers [71] and [51] referred to the evolutionary relation between requirements in different versions. A feature model or a design element could be evolved as mentioned in the paper [63]. Evolution type could be used to describe how the code base is evolved at each iteration as written in [61].
3	Refinement	The development team progressively refine the need statements of customer into solution requirements as mentioned in [97] and [51]. Refinement was used in the paper [8] when following the tracing link between a requirement to a code component.
4	Contribute	It is necessary to track who contributed to the user story by creating a feedback loop to keep this stakeholder notified of progress as written in [79]. Annotations and comments on many Story-Wall could differentiate contributors of these stories [50]. This type was written in [90] to tell who was the requester for this requirement.
5	Rationale	As described in the paper [16], this type of tracing relation could help developers answer questions what is the purpose of this test case? and why it has been implemented in that way?. Rationale was mentioned in [97] when we tracked the link stating why a design document has been documented from requirement specifications.
6	Conflict	The paper [63] elaborated that some executable acceptance testes can get passed if features are selected separately, but fail when putting these features together. That means there is a conflict between these features and we need to track this link.
7	Satisfy	Satisfaction was mentioned in [90] when tracking the link to see how a requirement has been tested and improved to satisfy stakeholders.

2. **Results from interviews:**

- *Results of the traceability Matrix:*
  Before summarizing what types of traceability are actually important, interviewees filled in the traceability Matrix to express their opinions. Totally, 18 specific links in the Matrix (see in the Appendix) were used as examples and then were matched with relevant types among 08 types mentioned in section 2.1.7. The Table 7.9 demonstrates the number of tracing links that each interviewee named and matched.

  According to the selection of interviewees in the Table 7.9, ‘Rationale’ had the highest number with 116 links. Next came ‘Contribute’, which had 99 links. ‘Refinement’ and ‘Dependency’

Table 7.9: Results of the traceability Matrix

Company	Dependency	Refinement	Evolution	Satisfy	Overlap	Conflict	Rationale	Contribute
A	2	12					12	10
B	8	8	5	8	1		8	10
C	8	4	4	6	1		10	10
D			10				12	12
E							12	12
F	5	8					10	
G	5	9	10	3		1	5	5
H	3	12	8				12	10
I	10	5					10	5
J	8	2	4	7	1	1	10	5
K	6	7	3	7	1	1	5	10
M	8						10	10
Total	63	67	44	31	4	3	116	99

had nearly equal number of links, 67 and 63 respectively. Practitioners matched 44 links with 'Evolution' and 31 links with 'Satisfy'. 'Overlap' and 'Conflict' were filled with only 4 and 3 links respectively. We could visualize the results of the traceability Matrix by the column number in the Figure 7.9:

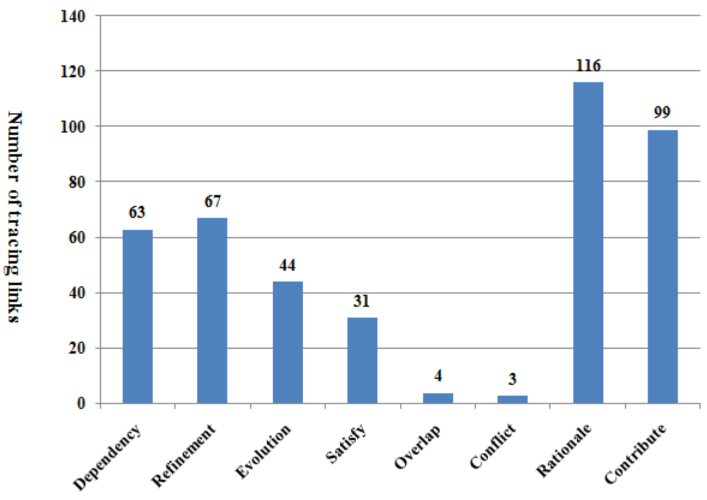


Figure 7.9: Numbers of links and their corresponding types

Or by percentage in Figure 7.10:

- *Interviewees summarized what is actually important for traceability:*
  The traceability Matrix was based on the input extracted from the paper [32]. However, some interviewees were confused with this Matrix. Thus, after filling in this Matrix, interviewees either used the types in the Matrix or they used their own words summarizing the important types of traceability. The Table 7.10 briefly shows what is *important* and *not* important traceability according to the practitioners:

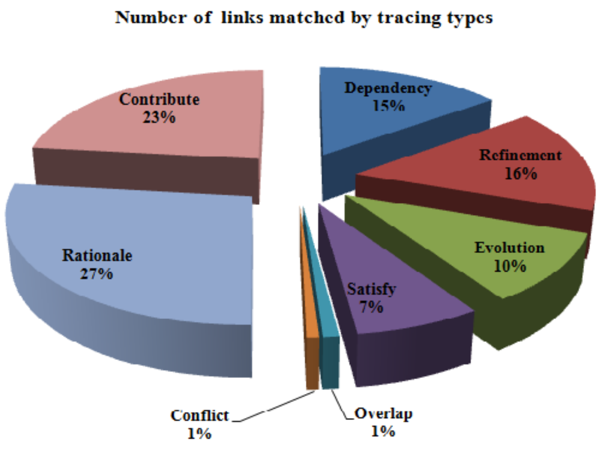


Figure 7.10: The importance of tracing types and their percentages

Table 7.10: Interviewees described what is important traceability

Company	Important traceability	Not important
A	Full traceability chain from high level requirements to detailed requirements to test plans/test cases and also to implementation stories in the product backlog/sprint backlog is of main importance. The reports (statement of compliance) clarifying which requirements that have been implemented. We are also are setting up suspect links handling in the applicable ALM tools. This means that if a requirement is changed, a defined profile set to what type of linked artifacts (requirements) that need to be verified if they are affected. The tools always trace who has written a requirement, story, test case or made which change etc. So, from high level requirements to detailed requirements; from detailed requirements to test plans/test cases; form detailed requirements to implementation stories in the product backlog/sprint backlog. Rationale is covered by the Vision statement in development projects based on Scrum, or a Roadmap for a knowledge area etc.	Overlap and Conflict are party handled today where the requirements are made up of models or by using GAP analysis or a graphical representation of all requirements for a specific product or release. It is handled differently product by product or even feature by feature.

B	I worked for 04 companies. Some companies are strict for processes, requirements, documents, specification, etc. My current company does not release any software, we do not have many requirements. We just go ahead, starting everything from requirements to go all the ways down. We cut short all those steps and we move very fast. We come up with ideas, try to come with the ways or solutions. Importantly, if you spend much time with requirements, documents, all kinds of test verification, it will take a lot time. If you go ahead with minimum requirements you can throw out a good idea to the public, they try it. If they like it, you get feedback immediately. So it depends on situations, some projects require very good documents, e.g. long-time project require documents such as design, verification, etc. That mean you have to plan upfront for all tasks.	For short projects, for marketing purposes, we ignore many documents, we just go ahead and build them quickly and throw them out to the market.
C	Apart from what I filled in, I want to add the relationship between the capacity and work in Agile mode is important. You trace between flows of requirements or design to what you actually take during each Sprint is very important. When working in Agile mode, you cannot know exactly levels of details of work you are taking during a Sprint because it can be changed, Right?.	
D	It is essential that one of great things about Agile methods is you get rid of lots of software that do not really contribute to. Being able to maintain code and constantly re-develop it and make them nimble and faster and management, you need to know why certain features were there from beginning. That is you did not remove essential functionalities when you simplify system. Knowing why you do something is really important. It is good that Agile helps to get cores of what customers need and why because you involve customers. In short: it is very essential to know who adds what and why ?.	I would say the small thing that you discover in the automated testing. We do not care since we can use tools to fix quite easily.
E	I would say you need to trace everything, from business activities to client. You need to show what has been done and why. Now, I have a co-worker having big Agile project where the customers change requirements over ... over ... and do not know why the budget is not same as at the beginning. They totally re-did every thing, I want to know for each user story what is the exact cost? If you do not keep track of traceability, you cannot motivate all things have happened within the project to talk to your clients. You need to keep track and trace who adds for what and why? How much resource for this change ? What decision has been made? Why?, keep track of things like what the customer actually said to me. If you do not have papers tracing on it, you may get trouble later on.	

F	In some TDD-frameworks, you pick a requirement, if it's well written, just split them into fractions. All tests are written before the implementations and test cases/scenarios are directly based on requirements. You will have a very nice documentation of what the system is expected to do. When skimming through your own code, you will not be confused why is this here.	
G	I think it is important to trace from requirement to some kinds of verification. When you have requirements, you are able to do test cases, verification and verify that requirement in sections are fulfilled and I think that is the most valuable. You need to trace from problem report to verification as well. So everything you decided that should be done in project no matter of which kind artifacts. You are able to trace down to verification. Within project, most likely to trace from requirement problem report to deliverables, what requirement is delivered? when? and where?. It depends on project by project to consider each situation separately when thinking about what level of traceability you need and what kind of traceability is most important to you.	Tracing down to line of code, test case, etc is too expensive. If you have good tools, it could be valuable to trace details. However, we do not care this issue.
H	We focus on 'Refinement' type from customer requirements to system requirements, then to subsystem requirements, down to system elements. Then we have 'Rationale', so we think important to describe why you change something?, why you implement in a certain way?. I think evolution is also important because to transfer knowledge from one engineer to other as you can make decision on something. Satisfaction is important as you have to satisfy requirement and design at upper level.	Conflict and overlap are not cared as you need to do before tracing. We use descriptions in design documents, communication between engineers keeping knowledge in heads to share, discuss how the system working, what depend on what.
I	We have system requirements and have them broken down to subsystem requirements or tasks. User story is telling what features need to be implemented and why?. Importantly, team members always discus as it is very often that user story changes. So we know who changed what and why?	We do not need to update depended connection because we have system architecture specification tool to fix this issue.
J	Probably requirements involving the product owner because that is very important. The product owner or customer needs involve much in works for tracing from stakeholders to requirements. Various stakeholders contribute. Obviously, customers need to contribute to requirements. For design, it is architects, developers or both. They need involve in that. Actual code, if you do unit test, should be tester and developer, coder involve. Others artifacts involve document delivers and could be users of subsystem, other people involve, I mean a lot of stakeholders need to involve.	

K	For tracing from requirement to other artifacts, the requirements should satisfy all relation types among requirements because the requirements should be clear, exact, and detailed enough for Design, implementations and Test to follow. So satisfaction is important as it could block the development process.	The overlap type is not so importance since it does not block the development process.
M	Refinement is usually mandatory, if you not do that your requirements would not make any sense. Depend on methodology for expressing requirements, you have wordings and put in Agile context. You start out with systems, you will refine them with user stories or items. I also need to know who where this user story has been implemented like this? who modified this and why . Talking to satisfaction, you usually trace from test back to requirement is important definitely. You will have clear picture for criteria satisfying requirements. So it is important.	Evolution, Overlap or Conflict (hmn...), I would not think as important and put too much effort on these types as I would reply on business analysis or whoever.

3. **Combining interviews and literature:**
   From this point onward, types of traceability exactly refers to types of tracing links or relations. It does not refer to very general types of traceability described in the sections 2.1.4 or 2.1.8 of the Background chapter. Descriptions of interviewees in the Table 7.10 above were interpreted and reasoned to summarize main ideas, feedback or concepts. Then they were synthesized with total number of each traceability type in the Table 7.9. Synthesized findings of interviews were merged with findings of literature in the Table 7.8 to make the Table 7.11, on which the Analysis section was based.

Table 7.11: Summarizing findings from interviews and literature

Types	**Industry interviews**	**reviewed literature**
Dependency	Only company A directly mentioned this type. Totally, 63 links of the traceability Matrix were named 'Dependency'	06 papers discussed this type.
Refinement	Companies H, A and M discussed this type. Additionally, 67 links of the Matrix were named under this type	03 papers discussed this type.
Evolution	Company H directly mentioned and 44 links of the Matrix were matched this type.	04 papers mentioned this types of traceability.
Satisfy	Company K mentioned and 31 links of the Matrix were matched with this type	01 paper discussed this type.
Conflict	No company mentioned and only 03 links of the Matrix were named under this type	01 papers reviewed mentioning this type.
Overlap	No company mentioned and only 04 links were named with this type.	0 paper discussing about this type.
Rationale	It was mentioned by most interviewees. In addition, 116 links of the Matrix were matched with this type.	02 papers discussed this type.
Contribute	Companies A, E, F, D, I mentioned and 99 links were matched with this type	03 papers discussed this type.

4. **Analysis and discussion of this question**

4.1 **Dependency:**

According to the results of literature, dependency was the most important. However, only one company directly mentioned this type as shown in the Table 7.11. This *difference* could be explained that the size of projects is different between actual projects of interviewed companies with small projects or experiments reported in literature. According to the interviewees, their projects were quite big and had so many depended connections. They could not spend time to pay attention to the individual depended connection. Therefore, they need effective ways to handle depended links quite easily. For example, companies D, H, G and I told that they are using communications within the development team as the best way to track the Dependency. Interviewees also supposed that many available software tools could help to track depended links quite easily. On the contrary, experiments in [79, 23, 89, 67, 63, 128] had few depended connections, to which researchers paid attention.

Another reason explaining for this *inconsistency* could be that traceability mechanisms reported in reviewed papers were mostly manual while companies are using automated or semi-automated traceability tools. For example, Vassilka et al [89] reported that a simple spreadsheet could work well to trace depended links in small projects. Or Abdallah et al [128] wrote that many traceability approaches did not explicitly maintain the depended link between unit test and code class under test and those approaches were mostly manual. It could imply that when using manual traceability approaches, dependency links may be considered as the important type. However, though Jacobsson in his related work [79] stated that tracking the depended links was necessary, he supposed that in Agile development this type of links could be handled quite easily since links between artifacts may be iteratively updated and could be done by many available tools. Therefore, it could be deemed that practitioners in Agile projects will probably not consider the Dependency as the important type.

4.2 **Evolution:**

The results of reviewed papers in the Figure 7.8 show that Evolution was mentioned as the important type. However, this is *not consistent* with findings of interviews in the Table 7.11 where only company H directly mentioned the evolution as the important type. This *inconsistency* could be explained by several reasons. To begin with, the evolution link was considered as the important type in literature [133, 120, 121, 105, 57] since it was discussed in the traditional software development process, in which there seemed to be a default for the evolutionary development process of software artifacts. In Agile, on the contrary, Espinoza and Garbajosa [9] wrote that the evolution type should be defined according to characteristics, needs of a specific project and requirement artifacts need to be subjected to iterations. It could imply that there will *not* be a default to define the evolution of artifacts since practitioners need to define this type of links, depending on each individual project.

According to feedback of interviews, practitioners in each Agile software project need to go back and forth to see the evolution links between requirement, design and code artifacts. Interviewees stated that they could track this type of connection quite easily. It could mean that since this task needs to be conducted very often or daily, so that practitioners should know effective ways to keep tracks of evolutionary connections between artifacts. The evolution type, therefore, seems to be less important in an Agile project.

4.3 **Rationale:**

This type of traceability seemed to be the most important type according to practitioners. During the interviews, most interviewees mentioned the reason why something was added, made or implemented in a certain way. For example: company D stated that "...You need to know why certain features were there ...". Company E stated that "... why this decision was made?...", company H expressed "... why you changed something? why you implement this in a certain way". In addition, interviewees matched 116 links, accounting for 27% (in the Figure 7.10) 'Rationale'. Though only few papers discussed 'Rationale' as shown in the Table 7.11, it does not imply that literature paid less attention to this type of traceability since the number of primary studies in this thesis is very small.

More importantly, findings in literature were in line with feedback of practitioners. Ramesh and Jarke [133] stated that a major challenge of requirement engineering is the linking of *rationales* and sources to the requirements. It could imply that Rationale links between requirement-related artifacts and sources need to be specially taken into consideration. In addition, Sukanya et al [129] stated that team developers want to know the purpose of a section of code and test case. That implies the 'Rationale' connection could provide developers with clear information explaining why this code or test case has been implemented in that way. Therefore, knowing why something has been created or done in a certain way could be really important to all stakeholders involving the development process of a system.

4.4 **Contribute:**

Contribute seemed to be the second most important type. It was confirmed in transcripts of companies A, D, F, E, J and M where the word 'who' was repeatedly stated. Company A stated that "... who has written a requirement, story, test case or made which change ...". Or company D said" ...why it was there and who made this, e.g. Peter said that he wanted this, thus he wrote this in code...". Interviewees selected 99 links of the traceability Matrix as the 'Contribute' type, accounting for 23% of the total links examined. Findings of interviews were in line with insights extracted in three papers as illustrated in the Table 7.11. Particularly, the important paper [90]-which actually reported results, feedback from 05 interviewed companies, wrote that "... who is the requester or stakeholder as well as from which process stems the requirement...".

Furthermore, Gotel and Finkelstein [57] stated that tracking the link between people and requirement artifacts is particularly important since this provides the firmest foundation for dealing with many issues related to pre-requirements traceability. Thus, it could be concluded that knowing who did what and why he did so? are obviously important because not only do these questions help us to track the link between a stakeholder and his responsible artifact, but these questions also help to identify problems, i.e., where bugs or errors come from and why?.

4.5 **Refinement:**

This link was seen as the third important type. Companies A, H and M put this type in the top list of important types. Company M stated that refinement is mandatory while company H said that "...we focus on refinement type from customer requirements to system requirements, then to subsystem requirements, down to system elements ...". Very often, interviewees described the process to define tasks, how break down systems into subsystem, how to combine existing elements to form new elements. So it could be implied that refining a software artifact i.e, a big source code module, and then formulating and elaborating this big module into several small code components is obviously. During the interviews, interviewees named 67 links of the Matrix as 'Refinement' type. In addition, findings of literatures also supported findings of interviews. For instance: in papers [87, 120], *Refinement or Generalization* relations were used to represent logical entities at different levels of abstractions between tracing requirements. Furthermore, papers [61, 51, 97] discussed and supposed that this type of traceability should be important.

4.6 **Satisfy:**

Results of interviews and primary studies showed low interests in this type. This type of traceability may be seen as the important in traditional software development. For example, the satisfaction link in [133, 177] between a requirement and a design component, in [121] between two requirements or in [2] between two user cases. This could imply that during the development life cycle of a system, a team member needs to make an element to satisfy or meet the expectations of another element. Otherwise, he will not finish his task and go on. According to interviewees, however, the satisfaction in Agile development process could be seen as a dependency and it is not considered as the important type because a team member must know the pre-conditions of an artifact. Therefore, it could be assumed that team members will not consider this type of traceability as the important type.

4.7 **Conflict and Overlap:**

Findings from both interviews and literature showed almost the same facts (1% in both Figures 7.8 and 7.10) saying that these two types could be ignored, even no company in the Table 7.11 mentioned these two types of traceability. This probably means that these links could not be concerned since they could be easily tracked and handled. For example, companies A and M told that Conflict or Overlap could be easily conducted through gap analysis or requirement analysis. Company K said these types are not important because they do not block the development process, while company H stated that these types should be inspected before doing traceability. This means it is obvious and necessary that team members often need to check if there is any Overlap or Conflict between software artifacts.

7.2.2 RQ2: What mechanisms are used by Agile companies to achieve traceability?

Aims: To identify what tools, methods, approaches, techniques or frameworks that are used by companies to achieve traceability. Through this answer, we also know why some other tools are *not* used in these companies.

1. **Results from interviews:**
   Interviewees briefly described what traceability mechanisms are *used* and what they do *not* use in the Table 7.12. After that, these descriptions were interpreted, reasoned and summarized with the aim of supporting the Analysis section.

Table 7.12: Traceability mechanisms are *used* and *not* used in companies

Company	USED	NOT used
A	We are using the same tools, based on IBM Rational JUST Platforms. Server based tools, web based client for most tools, you have common log in for all tools, you get 03 main tools. RRC(requirement management tool); RTC (tools for product Backlog management, CM part to handle code); RQM(test management tool: all test plan, test case, automated, so on).	Open sources, JIRA, MARS, team foundation server ReqPro are not used because we aim reducing costs. But we are freely to select tools so long as these are effective. I know test management tools made by Swedish HanSoft and this tool needs manual integration.

B	Not really because one requirement of our system is flexible for change. We just do manually, not need tools. It really depends on levels of your team members. We have much experienced members (20-30 years), we know exactly what to do. Something is obvious. When ideas come up, they know very quickly what need to be done as they have much experience. They know how to turn things quickly. They need not to spell out; they cut short a lot, to get things out. So it depends on situations that we make decisions.	I used to use IBM rational, Clear Quest. I did evaluate some tools last years for change management or requirements. It depends on your managers also. If he is technical guide, he will care about process, framework and invest resources on process, tools. If he is not, he just wants something very quick. We are not buying tools as we are much experienced, we know what exactly need to do as quickly as possible. Additionally, we can use a lot of open sources. It totally depends on project by project and depends on individuals to figure out how to solve problems.
C	IBM rationale, team foundation server, Scrum backlog. We could not use tools fully because of some bugs, and lacking of knowledge. So we have tools and use tools some specific degree, but you can manage dependency and kinds of relationship among different objects: product backlog items, Sprint Backlog items, which can be traced. So if you have any softer, then main tasks automatically start. So that is level of details you can manage by using particular tools, but we could not use these tools because of some other commitments and time plan.	A lot of tools within the group. For dot Net and MS area, the team foundation server should be used. For web or java area, I don't know may be IBM rational. I think this tool work well but rather complex actually. To be honest, we do not use this fully because of lacking skills and we do not have time enough. I think that we need to improve this situation.
D	We used tools for traceability. I am not sure I remember, but tools are not from big companies, they come from small companies. We use a lot of open source, we also bought traceable software. We looked at IBM rational. The thing is that since a few years back no single tool could handle all.	I know IBM Rational, Clear Quest but we did not use because they are not good enough for us. That is why we use open source. At my time, these tools were good for really big project, but it was cumbersome for our size, it cost 10 million SEK. So it was impossible for us. The point is the balance of costs and benefits.
E	We use some models of Agile, e.g. product Backlog, MS. excel sheet, email. We also use a specific tool like JIRA for handling all aspects of requirements. Some of my members use team foundation server internally.	IBM rational, Clear Quest, SHARE POINT But some teams at my organization may use other tools as we are big Consultant. But my big clients, e.g. Ericsson, are using these tools. I cannot use all tools because I need to choose among tools, so we do not need duplicate to save resources.
F	We have one tool to keep track of requirements to see when they are delivered to customers. You can set status for each requirement (finish, done, deliver, accept, reject, etc). For other thing, no, it is the only tool for managing requirement. The tool is name Pivotal Tracker. If something is changed, maybe we have time to update. Most of things are specified in Excel. They do not have links between them. They said this feature need be done for something. If they need update for change, a member knows how to fix by himself.	I do not know other tools for traceability. But I heard about Kanban Board, where you post notes, you write requirement on board, and you write on columns (status of artifacts). You move around, it is kind of same idea. I think systems like this there are some. But I think they do not help traceability really you miss these links. You cannot see how things are connected.

G	We have some scripts that help developers, we do not buy frameworks for doing traceability. We do traceability with our own built tools, adding methods, scheduler tools, to build systems that create traceability. The rest is mostly self-developed tools, something like that and enhance the environment to create traceability. We are using Clear Quest tool for problem report handling. But traceability tools are developed by ourselves for gaining the information from the problem report tool... So it is more we reduce database, I mean information is stored in Clearly Quest but the access is self-developed.	A lot tools, (big laugh), Scrum Work, GRAP, REMIND, Clear Quest, RepPro, Rational Team Concert IBM, IBM bug, a lot ... The main reason these tools do not work for us is we need to develop by ourselves for our own needs. I think that is part or totally that there is strong culture for developing solutions by ourselves rather than buying. If you buy tools, requirements are both installation and training cost, maintaining cost, support costs. I think that we are feeling you do not gain that much from buying tools. In many cases, you can do really slim solution by yourself.
H	We used to have dimension RM. We also use DOOR for handling requirements. We have tools, but we are on the way to be better, not all engineers are using these tools because tools are complicated to learn and use. Now we are trying to simplify the use and to get better GUI, versions of tools. These tools are beneficial to our company.	RepPro, IBM Rational, Focus Point. We might be using Focus Point for some requirements when you select most important requirements for priority before they are actually put into project for comparison between requirements. I think we did comparison between tools and we made decisions basing on different factors to select which tool is the best for us.
I	This tool called AGILO tool. We are mainly manual as the size of the company is quite small. So we only need AGILO to support for handling requirements. We also use Scrum Board, Wiki. The rest is communications within the team.	MARS, ReqPro, I heard some other tools but not remember. There are pros and cons to select tools. It depends on some factors such as: resources, projects, etc. I assume no perfect tool.
J	We have ReqPro Repository, Issues Tracking System , MS Visio Studio, team foundation server are main tools	We had used Scrum Work before, e.g. Scrum Board in Agile. But it is not pretty good. So we used other tools instead. They do not work for us, in a small project they may work. But we are too big projects, thus we need tools to control our traceability tasks.
K	Scrum Work, JIRA, GIT, some open sources	A lot, cannot remember. It depends project by project actually.
M	Presently, we use excel sheet for some extend. Then you put in systems such as: MINGO, Clap, JIRA, Repository version control. We use IBM rational for big project. We have ReqPro, Clear Quest for tracking requirements. I used TRACK, Wiki for several projects before. We used MS Visio Studio, Team foundation server in some projects.	Well, RepPro tools in database for tracking documents. Looking at my organization, I will like use MINGO, but it is expensive. JIRA is also expensive for us. Some tools are mandatory within the organization. So you can not buy more other tools.

2. **Analysis and discussion of this question**
   As seen in the Table 7.12, companies reported different tools they are using as well as many tools they are *not* using. The most common and simple reason-according to interviewees, is that it depends on a specific project. That could mean after defining the level or the needs of traceability, companies will decide which tool should be used. Most interviewees stated that they are using traditional tools such as: excel sheet, product Backlog and white board. Requirements need to be broken into items,

which are stored in the product backlog with excel sheets before they are moved to systems of tools. It is, therefore, assumed that without the product backlog and excel sheet, it may be very difficult to conduct traceability since these traditional mechanisms may be very effective ways to implement traceability. Besides, commercial tools made by Microsoft or IBM are used in big companies. For example, global consultants like J and M are using Visio Studio and Team foundation server. Global companies like A and C are using IBM rationale (ClearQuest, DOOR). Companies E, K and M are using the tool named JIRA while companies G, D or F are using self-developed tools or open sources.

However, it seems that there are two tendencies. For global companies like A and J, they are willing to buy commercial tools to do traceability. The reason could probably be that these companies may have very big and long-lived projects, which need to be broken into subsystems and distributed to many teams. Therefore, global corporations may prefer to provide commercial tools for their member companies to use. On the contrary, small companies like F or K or medium consultant like E, prefer to use self-developed tools or open sources. It could be explained that these small companies have small projects and their team members may have good capabilities to make and combine open-sources to set up self-developed tools satisfying their actual needs.

Interviewees stated that using a tool or not, which specific tool to use depends on many criteria. The essence of the matter could be the balance of costs and benefits. Companies D and G stated that they prefer to use self-developed tools because these mechanisms are much more flexible. It could be deducted that self-developed tools may be easy, suitable for small size teams with short-term projects in small companies. Big companies like A, H or C who bought expensive commercial tools from IBM or Microsoft, may think that they are very big organizations with hundred of offices or subsidiaries around the globe. So buying one or several commercial traceability tools and then having them used within the whole organization may not be expensive. However, buying a commercial tool like IBM Rationale seems to be impossible for small companies since it could cost too much money. Companies D and G stated that not only do buyers have to pay for tools themselves, but they also need to pay for supports, updates or trainings services. This could imply small companies may focus on self-developed tools to conduct traceability with the aim of saving much money.

Interviewees provided some reasons why many tools that are not used. First, the tools may be complicated to use as mentioned by C, H or tools could not satisfy the needs of companies according to companies D and G. Second, companies probably try to save costs by re-using old traceability mechanisms. For example, some companies said that tools supporting traditional software processes could still be re-used to support traceability in Agile. However, this is *inconsistent* with findings in papers [61, 51, 22, 13], which reported that traceability mechanisms characterized with any traditional software process have shortcomings and cannot adequately support the Agile development process. This *inconsistency* may be explained that a big corporation like A, needs to break down a big and long-lived project into small projects or subsystems, which may be developed by using traditional software methods. Most companies stated that they are mixing Agile, RUP and Lineal together. This could imply that with the aim meeting actual needs, the combination of software development processes should be applied since it may be impossible for a company to be fully or 100% in Agile, no matter what size the company is.

7.2.3 RQ3: What mechanisms work well in Agile development organizations?

Aims: To discover how companies are using their mechanisms? and to see if these mechanisms work *very well* in Agile companies?

1. **Results from interviews:**
   In the Table 7.13 below, companies described how their traceability mechanisms work well in Agile software projects. After that, these descriptions were interpreted, reasoned and summarized with the aim of supporting the Analysis section.

Table 7.13: How mechanisms work well in Agile companies

Company	How to use mechanisms	Feedback or Evaluation
A	From main requirements, divided between different products, you shoot down requirements to specific RRC project areas. This requirement project area either based on project or product or feature, etc. From this requirement project area, you can have several teams working. So they could either be set up as several teams as one RTC Backlog area ... or connected to same requirement area. That is the same for test management tool. You could either have 01 main requirement to be traced to 01 requirement and traced to a backlog item, test case. Or you have collection of requirements connected to the test plan, you can get statement of compliance saying for requirements (Statement of Compliance and Statement of Verification).	These tools are beneficial to our organization. Different options, but this is set up for us. But our department just uses IBM rational and this tool is beneficial because we are not co-located in one place.
B	It depends on project by project and much depends on individual experience. I could, thus, say no specific tools are mentioned.	We are working very effectively. Each member needs to find his own effective way to handle things basing on his experience.
C	You have different templates that you can actually use to clear the product backlog items. When you come up details for implementation, product backlog items can be broken down into Sprint backlog items related to a specific Sprint. When it comes to Sprint backlog item, if you have a dependency traceability to some another item to complete, you can define impediment and different types of levels. When you define impediment or dependency at different levels, when you complete this impediment, you can come back to the previous author, who is responsible for a particular task.	I think this tool work well but rather complex actually. To be honest, we not use this fully because of lacking skills and we not have time enough. I think that we need to improve this situation. Without the product Backlog, it is impossible to have traceability because you need to be clear, what exactly you are doing because a lot of things during the project also. Having Backlog at the beginning, it will reduce specific scopes or also give Scrum master specific tools or probably ... to control that scope of the project.
D	At that time, we picked some part of this tool, part of other tool and part of other tool, etc. We had to combine tools to support our needs. It is essentially that we had the database called file maker for tracking and managing requirements. We put different functionalities and added them on top of this database.	We made these tools to work. We put efforts making them work for us. We bought part of functionalities, we used some open source. Then we combined for what we want or need. Thus, I thought these tools work well for us in Agile.

E	When it comes to handling Backlog, it is more formal thing about specific requirements, we use those kinds of tools to customers to make priority and keep tracks. You need to keep track of requirements somewhere.	Depending on projects to decide using tools or manuals. We also said some tools are not effective with some clients. If I look back to my background, my managers told that you should use for this tool for handling requirements, this tool reports, this tool for communicating, for change request, etc.
F	With this tool, you can update for each item/requirement. It is a webpage tool, so customers can access the system. Thus, they can see our progress when they check it out. When there is change, we discuss in team. Basically, first you create specification for system, you decide things, you break down requirements to items. You add to system backlog, and then you write in Sprints. You start working with assignments for developers, who are assigned by managers. If any change in these, customers access the system to change. I will see what has been changed, and then understand what I need to do.	I like this tool as it has nice Interface, it is easy to access for both developers and customers. You get notifications by email if any change in your task. So it is good way to manage your work. This tool does not depend on Agile practice or traditional method. It just is a way to organize things to keep noted about what should be done, like “to do list”.
G	As mentioned before, depending project by project, we will develop traceability tools by ourselves. We use Clear Quest to store information and use self-developed tools to access information/data stored at Clear Quest.	These tools are benefit a lot. We do not have norm, time to spend on tools, but it increasing quality week by week. We are doing continuous improvement for tools, so it is getting better and better for our tools.
H	We use the tool for tracing requirements between system level and sub-system level, and also to test specifications. That means we break down system requirements into to sub-system requirements and system test specification. We continue to break down into system elements or small items. Thus, you can trace from system level through sub-systems, down to system elements, etc.	Our tools work well with Agile practices. We have tools, but we are on the way to be better, not all engineers are using these tools because tools are complicated to learn and use. Now we are trying to simplify the use and to get better GUI, versions of tools. These tools are beneficial to our company. I think that DOOR has already been upgraded to 2014 for better. They are simple and they are suitable for costs.
I	When you start, you break down system into subsystem, and down to requirement/item or whatever you call. You put these stuffs into Agilo, which is the web database to manage your project actually. So you do a lot of things such as: see requirements for each task, search tasks, update status, break down tasks, burn down, etc.	My tool work well enough, something is quite slow sometime to work with Agilo. Not all tasks are demonstrated here in this tool. But it is enough for us.

J	These tools are very beneficial to my organization. We use initial tracking system, in which we put requirements, we create issues, we break requirements into work items, they are contained in tools. Everything you work with documents, artifacts are linked to these issues.	These tools are very beneficial to my organization because we have big projects. Other tools we used before, i.e. Scrum Board, not work well for us as they may fit small projects. Tools work fairly well with Agile practices I would say so. I mean you still use Excel sheet, white board also, go off line sometimes. I also think TDD is very good as you get actual artifacts or needs, documents, code. These help you have traceable system.
K	For each tool, you just break down your system requirements into work items, which you move and store in these tools that will help you most of your needs: tracking links, updating status, searching, etc.	So far Scrum Work seems to work well with us since we usually have small teams, 5-9 members for different projects. It's simple and can quickly adapt to changes from requirements and designs since the SCRUM's sprints are usually short, 2-4 weeks. On the other hand, GIT helps manage source code very easily and quickly.
M	You start with words, taking notes, and rapidly move the list into Excel. Those will be actual statements, goals, rationales. Then having requirements of product and writing requirement statements in whatever format. That would also be done in Excel. If you use tool, may be in case the project is large. Typically, you use Excel, you have sort of actual development projects. You enter all requirements into tool systems, then you use during the development. So you combine Excel with tools effectively.	These tools are mandatory for my team. Some tools seem not to works so well with Agile practices. I would do not recommend not using these tools, but we have to use them. But MINGO and Tell Low are working very nicely with Agile practices. Tell Low is very simple, MINGO allow you to involve in tracing requirement to design, and down to other artifacts, etc during the development process.

2. **Analysis and discussion of this question**
   Results of the Table 7.13 may state that in small companies like F, K or medium organizations like D, E or G, traceability tools are being exploited more effectively. Several reasons could be explained for this statement. First, companies may probably have analyzed the needs of traceability for each project before developing traceability tools. Thus, their tools could obviously meet the actual needs of companies. Second, the size of these companies is quite small, where a small team of members may work together for several small projects. Thus, team members may probably know how to quickly make tools or use open sources in effective and flexible ways. On the contrary, big companies like J or M may have big projects, which are assigned to a big team distributed at many locations. Thus, it may be difficult for team members to have strong cooperation to use traceability tools in an effective way.

   Though it was said that tools were beneficial, global companies like C, H or M admitted that tools are complicated and may not be working well. The explanation could be deemed that a global group decides which tool to buy and it is mandatory for companies within the group to use this tool. Another reason could be given that employees lack knowledge and skills to use the tool. When asked "what do you expect from the future tools?", many interviewees want a very simple, traditional tool like "nice virtual white board". In addition, tools should support for further collaboration, should be quite intuitive and flexible for new members to adapt to the process, new working environment. This feedback could be inferred that some current commercial tools may be really complicated to use since teams may not have their own rights to select traceability tools suitable to their needs.

Big companies told that they prefer using traceability mechanisms, which are easy to learn and just support the needs partly, do not need to fully or 100% satisfy all needs. It could be deemed that some tools may be rather complex when trying to cover many functionalities, some of which could be replaced by other, more effective ways. As mentioned in RQ2, traditional ways like Microsoft Office (excel, word) or manual ones like white board, could be very effective to track traceability links since many employees may have probably been using these traditional ways for a long time. Thus, employees may use traditional tools very fast and effectively to do tasks or solve problems rather than spend much time on exploiting all functions of new tools.

Traceability mechanisms or tools aim to simplify the way of working, to enable team members to work faster. However, it may be not the tool, but the experience and skills of employees that are essential to get tools to work efficiently. For example, when asked to suggest a mechanism, which supports traceability very well in Agile settings, practitioners suggested some practices such as: product backlog, continuous integration, collective code ownership, small release and Test-Driven Development or Test First (TDD). It is very interesting that TDD is strongly suggested by both industry and academia findings. For example, papers [13, 40, 51, 67] summarized that TDD could support traceability very well, while companies M, F, E and I suggested TDD as the best tool of Agile to achieve traceability. It could be inferred that these companies may have good experienced employees, who have been used to making systems by writing test first. In reality, however, some developers may be against with using TDD since they have been using test last-the traditional way, for a long time.

7.2.4 RQ4: What challenges exist for realizing traceability in Agile organizations?

Aim: To find out different and detailed reasons explaining why conducting traceability activities in an Agile software project is challenging the team development.

1. **Results from reviewed papers:**
   In the Table 7.14, 05 categories were defined. The column "Type of difficulty" is to group "Specific difficulties", which referred to each phrase or term describing about challenge of conducting traceability. The column 'Reference' referred to a specific paper, where the corresponding term or phrase was found. The column 'No' referred to the number order corresponding to each specific difficulty.

Table 7.14: Specific challenges reported from reviewed papers

No.	Type of difficulty	Specific difficulties	Reference
1	Cost effectiveness	Standard fashion require manual traceability	97
2		cost much time to frequently update tracing links	97
3		Managing links in big project is burdensome	97
4		Assume traceability as extra work	89
5		Take time and extra effort	63
6		More work and additional documents	125
7		Very expensive	11
8		Heavy routines	11
9		Administrative overhead	79
10		It requires much time to learn many tools in large organization	89
1	Knowledge & Experience	Team members lack experience	39
2		Team members lack experience	63
3		Difficult to select the right degree of traceability	89
4		Difficult to use many tools	90
5		Defining available information is difficult	67
1	Cooperation, regulation or organization	Lack of strong cooperation from customers	40
2		Complexity structure of organization	89
3		No one reads documents	79
4		Missing documents to be traced	125
5		Members are resistant to traceability	13
6		Lack of motivation to do traceability	79
7		Organization disciplines and commitment	63
1	Technology or Technical	Lack of uniform among tools	89
2		tools do not meet needs of organizations	90
3		Current tools mainly support traditional software, not Agile	61
4		Doing Traceability cannot be fully automated	128
1	Others	Requirements are not fixed before development begin	115
2		Requirement traceability matrix is incomplete	67
3		Refactoring code disappears old and creates many new links	100
4		Conflict opinions about the importance of traceability in Agile	84
5		Cause developers loose creativity	11

The Table 7.14 shows that "Cost effectiveness" was mostly mentioned with 10 phrases. Next came to "Cooperation, regulation or organization" with 07 terms. "Knowledge & experience" and "Others" had the same number mentioning with 05 terms for each category. It was found that 04 terms mentioned to "Technology or technical" category.

These specific difficulties could be demonstrated by the number of terms found in papers. For example, the Figure 7.11 shows that 10 terms accounting 32% of the total terms in literature mentioned difficulties in association with "Cost effectiveness":

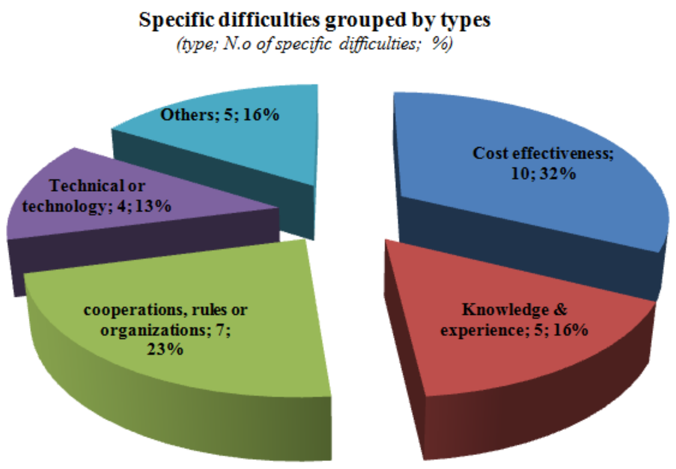


Figure 7.11: Types of difficulties reported in literature

2. **Results from interviews:**
Companies described what difficulties that they may face when conducting traceability. Interviewees also suggested some solutions to cope with these difficulties as shown in the Table 7.15

Table 7.15: Difficulties and Solutions for conducting traceability in Agile software projects

Company	**Difficulties**	**Solutions**
A	If you have too many dependencies, it tends to be large that is hard to get overview of traceability. That could be a problem actually. But main route because is often you slice deliverables in wrong ways. You will have to go back and read up for them, to get simple traceability, dependency between them.	We have service responsible within a team. They are responsible for this area to make sure traceability is added. We are trying not only cross-function in a team, but cross function individual as well. So that we do not have anyone assigned in the team that you are doing all the test things, this is not the case.
B	It creates overhead, heavy routines to make documents and implement traceability activities. The knowledge and experience gaps among team members are also challenging.	We use communications, e.g., sit down and talk a couple of hours, then team members get ideas and implement quickly.
C	Some difficulties: the self-motivating/organizing team; the commitment of team members; knowledge and experience gap among team members; difficulty to monitor progress of members, to control their commitment when having big teams distributed in 30-40 sites.	First thing, in Agile the meeting at the end of Sprint, it actually helps to understand, self evaluate team members. It is important to have clear: what is the definition of done or completed for each task? So that members to know very clear at the beginning.

D	It is difficult on low level because you change a lot. Essentially in perfect waterfall-model project, you write traceability once, that is really easy on low level e.g., this is red color and this is code, that is easy. But it is more difficult in Agile because you change all the time when you talk to stakeholder, and then you change software. But on higher level, it is so truly to keep traceability, and it is easier with Agile because you got pass the gatekeeper.	Developers need to make details for traceability as they are ones to re-write code if they did not have traceability. You know that I told you that running around when you have a lot of stakeholders, you speak to the first stakeholder, he tells something, you speak to the second stakeholder they say something else, etc, you keep just running around, as you not keep traceability, so you will never finish. Thus, if you did not do traceability, you will have to work many more hours.
E	Adding overhead, taking time from producing. Sometimes, traceability is requested as deliverables, sometimes you need to document but main deliverable is actual software you are making. So it adds costs and overhead for team members. You need dedicated resource for members to handling communication and requirements, all things.	I think you need have good company practices, you need educate staffs to follow them, you need a good tool so motivate them to use . Example: from my experience, we had internal courses about the methodology, project management course to infuse team members with company culture where they understand importance of things, like what will be consequence if you have bug in the end of the process or in the production ? You need to infuse right values to staffs.
F	The challenge is time, it never makes room for it in budget. It may be impossible to explain with customer why you need it. Ex: we build web shop, customer have not so much knowledge about IT, when we say we need to keep docs, or maintain software and you will need overhead as compared to original plan, e.g. 20% time plan to make sure traceability activities. You need a lot extra time, so it is very expensive. Customers say why do not doing perfect from beginning? Why am I paying for these stuffs that I do not see any difference?	
G	Yeah, challenges are often almost kind of manual way to deal with change. Then, always information is get better liability because people enter information with manually, it cannot sure they are correct, you lack of trusts. If you do not give see the full picture of values of traceability, and then it is difficult, especially in Agile when you focus on delivering values. As long as you can provide good arguments, good overall pictures, they are less problems with adding traceability.	Solutions to cope with: to educate team members, reasonable explanation, you have to be honest, work with transparency.

H	I think to understand where you have put different information is challenging. If you have to trace between requirement, but you also have design documents. You have to put in different types of information in different places. You express what is the actual need from the customer and transfer to functions, so that every members know where information should be put, how you express things in different docs, artifacts in database because it is quite confusing if you try to add information in wrong places. So important is find information for tracing, especially for new members.	You should think how you put information. If you balance the number of requirements and size of documents to be traced, and how you refine the different levels of system, I think developers will see how they can update links and see what benefit they can earn. We have good developers and they also document, and do traceability. I think it is company culture. By introducing simple tools or simple ways of handling our requirements, e.g., it is easy for developers to see traceability actually benefits. You need to define the definition of done at the beginning of the project, I think members would have no resistance.
I	We have the issues that team members keeping information, knowledge their head, not written. I think every company having similar difficulties. It is just personal perspective to resistance, not the whole team. That is the challenges.	We try to minimize traceability by working in pair like pair programming. The whole team should be involved to work with user stories. So they should not get task for individuals, it is the team work. I think it is very important.
J	People often search Agile as they do not want to make documents, so on. But it is not right way to do that. Agile still need docs, saying what have done. You need to be able to get understandings what happening. They think more fun to write code, but you need to take care of your own mess up. I think some are lazy as they do not want to do their work. They think less doc mean NO document. There is big difference here about the understanding less docs. Even, some people have resistance to making docs and updating traceability links.	You need to motivate the team make docs, update test cases, whatever, as they make your life easier afterward. It is not issue as it needs to be done. I mean, but the definition of done, you need to say that what have done. People need to fulfill what they commit to do. You need to be aware of what importance for project, product, and not personal importance. Yes, it is compulsory, we agree definition of done. You will not finish until you done with docs as docs belong to definition of done.
K	One difficulty is for R&D or prove-of-concept projects, usually it is hard for developers to estimate the time for each tasks. Because for such tasks, it is usually that developers haven't had much experience with. Hence the sprints are harder to maintain. Another difficulty is to have an experience team in which all team members has fairly similar knowledge level of almost every parts of the project's implementation so that one can switch tasks when needed. So it takes some time for team members to gather necessary knowledge and work together effectively.	Defining *done definition* for each team. For example, for developers, *done* for a task is when: - Verify the implementation resolves the task e.g. Functional test. - Provide test script (if any) to reproduce/verify if the task is resolved. - Get the implementation reviewed. - Commit the implementation to source code version control. - Update task's status in the issue tracking system e.g. JIRA.

M	It is not easy to explain the process to team members to understand how traceability to work for requirements, how develop tasks. That is the challenge; it is center that explains the traceability in Agile process. You need business analyst working requirements, helping customer to express their need ...you need developer to understand how to implement traceability at the same time they develop functionalities. So the challenge is the explanation to every member to understand the importance of traceability. Expressing the need of traceability to developers is challenging for business analysts.	In Agile projects, we remove, reduce bad process, manual work, and make things more active. You need select good tools; you need to support developers, teams by getting very good tools. Developers will not find difficult to do traceability. Thus, selecting good, simple tools to make traceability easy is very important. You need goal, purposes, etc of traceability to explain to developers.

3. **Combining interviews and literature:**
Descriptions made by interviewees in the Table 7.15 above were interpreted, summarized and classified according to 05 categories of difficulty defined in the Table 7.14. Then, these descriptions were merged with findings of literature in the Table 7.14 to make the Table 7.16, on which the Analysis section was based.

Table 7.16: Combining the findings from interviews and papers

Type of difficulty	**Reported by company**	**Reported by literature**
Cost effectiveness	This type of difficulty was mentioned by companies: B, E, F and G	papers: [79, 97, 89, 63, 125] reported this challenge.
Issues-related knowledge or experience of team members.	Companies: B, C, E, G, J and K mentioned in their transcripts	papers [39, 89, 63, 90, 67] mentioned this challenge.
Issues related to cooperation, regulations or organization within a company	This type of difficulty was mentioned by C, E, H, I, J, K and M	papers: [13, 79, 63, 40, 89, 125] reported this difficulty.
Issues related to technology or technique	Company C mentioned this	these issues were found in papers: [79, 128, 89, 90, 61]
Other difficulties	Problems caused by other issues were mentioned by companies A & D	Papers: [79, 67, 84, 100, 115] reported

4. **Analysis and discussion of this question**

4.1 **Cost effectiveness:**

Issues related to *cost effectiveness* are usually mentioned when discussing challenges of doing any task. Most interviewees told that implementing traceability requires companies to spend more resources (time, employees). The amount of resources consumed is obviously different between projects. For example, company H revealed that it cost them huge resources to do traceability for the software system installed in an airplane while company F estimated less than ten hours for tracking links since he did a very simple commercial web application. It could imply that doing traceability for a security software system will be compulsory and need to be done at the very detailed levels while a simple system just needs very simple traceability level. Buying commercial traceability tools is another challenge for companies since traceability tools may cost much money and may be impossible for small and medium companies to buy as the interviewee D said.

Findings of literature were in line with those of interviews. In his related work [79], Jacobsson wrote that customers have to pay more for the product since team members need to do a lot of extra work. It was reported in papers [97, 63, 125] that updating traceability links by using manual mechanisms would require much time, particularly in a big project or it may require much time to learn many tools in large organizations [89]. Obviously, conducting traceability activities requires companies to spend more resources. However, considerations should be seriously taken to evaluate what will happen to the system if traceability is not conducted. It could imply that the balance of cost and benefits of doing traceability should be carefully made. Therefore, *cost effectiveness* should not be considered as the most challenging.

4.2 **Most challenging difficulties:**

Interviewees considered some issues as the most challenging. First, *knowledge and experience* within a team seem to be important. This was in line with findings extracted from papers [63, 39], which stated that team members lacked knowledge and experience to decide the level of traceability or to use some traceability tools [90]. The explanation could be due to the fact that new employees may have to learn to use many complicated tools in big companies. One of the solutions to cope with this challenge is to provide more trainings or to re-train new employees as suggested by companies E, H, G and M.

Second, *cooperation and commitment* of team members need to be considered. As two project managers of company C said: "...it is difficult to monitor the progress, control the commitment of members when having the big team distributed in 30-40 sites...". Obviously, this matter could be understandable since team members could not work together in the same time zone or same geographical locations. When discussing the solutions to track the progress and commitment of team members, almost all companies emphasized the phrases "definition of done", "full picture of traceability values" or "company culture" that managers need to explain or motivate team members. It could be inferred that some team members may probably misunderstand core values of traceability or why traceability must be implemented. Related work [79, 13] also reported that some team members were resistant to documenting, updating traceability links and explained that no one would read these documents. It could probably mean that the cooperation may be weak and team members may not understand the importance of documentation tasks.

Third, *Policies and resources* given to traceability activities seem to be important in order to motivate and encourage team members. This was supported by papers [89, 63], which reported that complex structure, regulation and policies of companies may burden to doing traceability. It could be concluded that difficulties related to: *knowledge, experience, cooperation, commitment and policies of company* could be considered as the most challenging because these issues are related to attitude, behavior, commitment and education of employees, which may require much time and effort to change situations or to get improvements.

4.3 **Technology or techniques:**

This type of difficulties was mentioned in literature [61, 79, 60]. This is, however, *inconsistent* with feedback provided by interviewees in the Table 7.16 where only company C mentioned this type of difficulties. The company C said that they did not use the tool fully since it was rather complex. However, this may not imply that employees at company C may be facing difficulties related to technology. It could be assumed that with the rapid development of technology and techniques in the field of software engineering, practitioners could try the combination of different tools to gain effective work, instead of spending much time on exploring new commercial tools fully. This could be the reason explaining why most interviewees did not mention this type of challenges.

Though it was agreed that current traceability tools are not fully automated or do not have sufficient uniforms, Abdallah et al [128] and Lars & Tobias [90] wrote that issues related to technology may not be serious since experienced team members could have sufficient capabilities to use the combination of tools. This was confirmed by companies B, D, F and K who emphasized on the knowledge of team members and available open sources. It could imply that practitioners in Agile projects may face no challenges related to technology if employees of companies have capabilities to explore and exploit many available software tools.

4.4 **Other types of difficulties:**

Some other difficulties were also mentioned when discussing challenges of adding traceability. For instance: company A stated that getting the overview of traceability is very hard in a big project since it has so many dependency connections. Or company D said that implementing traceability at low level may be tough in Agile because change occurs very often when talking to each stakeholder. In other words, when talking to each stakeholder, she or he want to change something. Findings in literature [79, 67, 84, 100, 115] also reported these problems when doing traceability in Agile methods. However, several solutions were reported to deal with these issues. For instance: companies A coped these issues by having services responsible to deal with traceability issues within a team. These types of challenges, therefore, seem to be not serious.

7.2.5 RQ5: What benefits exist in realizing traceability in Agile organizations?

Aim: To find out and consider various positive impacts or benefits that the team development could gain when conducting traceability activities in an Agile software project.

1. **Results from reviewed papers:**
   In the Table 7.17, 05 categories were defined. The column "Type of benefit" was to group "Specific benefits", which referred to each phrase or term describing about benefits of conducting traceability. The column 'Reference' referred to a specific paper, where the corresponding term or phrase was found. The column 'No' referred to the number order corresponding to each specific benefit.

Table 7.17: Specific benefits found in primary studies

No.	Type of benefits	Specific benefits	Reference
1	Tracking progress	Tracking the links between Stakeholder and Artifact	79
2		Tracking private tasks and progress of team member	79
3		Help to focus on customer	97
4		Measurement of project progress	90
5		Drive software development process	100
1	Improve software quality	Change impact analysis	23
2		Software maintenance	79
3		System requirement validation	39
4		Baseline reproducibility	23
5		Support the lack of formal requirement specification	51
6		Measuring test coverage	90
7		Better software quality	63
8		Verification against 3th party requirements	90
9		Key Performance Indicator for organization	90
10		Product conformance	23
11		Process compliance	23
12		Improve system verification	100
1	Save resources	Help members to do tasks easier	79
2		Rapid assessment on impact of change	97
3		Help members to find necessary information to finish tasks	129
4		Getting very quick feedback	40
5		Keeping all information gathered, organized and easy to find	84
1	Cooperation & Share knowledge	Increase understandings of customer need	40
2		Better communications between stakeholders	63
3		Increase collaboration among stakeholders	13
4		Share knowledge within organizations	89
5		Organizational learning	23
1	Others	Help break down high level requirements to detailed requirements	90
2		The foundation and mandatory for large, long-lived projects	90
3		Project accountability	23

The Table 7.17 shows that "Improve software quality" was mostly mentioned with 12 phrases. Three categories of benefits: "Tracking progress", "Cooperation & share knowledge" and "Save resource" were mentioned with 05 phases for each category. 03 terms were named with "Others" category.

These specific benefits could be illustrated by percentage for each type. For instance: the Figure 7.12 shows that the number of phrases mentioning to “Improve software quality” is 12, accounting 40% of the total phrases discussing benefits of traceability:

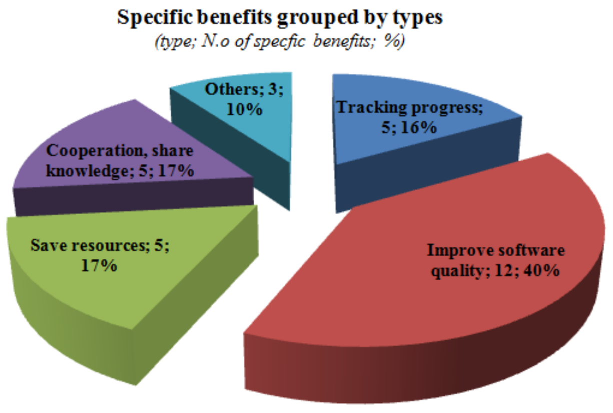


Figure 7.12: Types of benefits reported in literature

2. **Results from interviews:**
The Table 7.18 below briefly summarizes what interviewees said about benefits of doing traceability:

Table 7.18: Benefits for conducting traceability in Agile software projects

Company	**Benefits of Traceability**
A	Traceability not only fulfills requirements, but also important for marketing department when they ask to write marketing materials in new products because if you develop with Agile, you are not setting everything from start. Marketing department needs to know what is going to be changed, what need to be added. I would say that general opinion that traceability reduces needs of documents, so it benefits. Most people in program where I am working see advantages of traceability instead of writing lot of documents. If you see traceability as the leaving connection between requirements and use story, so on, you have tools supporting this you are working, you can easily get overview using tracing links instead of reading and writing documents.
B	It depends on your manager also. If he is technical guide, he cares about process, framework. If he is not, he does not care, he wants something very quick. Sometimes people coming up with ideas, they quickly build and throw out to get comments from publish. Depending on feedback, now I have to change. They keep track of change in requirements. They never have fixed requirements, just minimum. When you work with Agile, one thing you have to watch is all requirements will be changed because every time you read, you get feedback from customers. To keep track of change, it is very difficult. You just let the public to tell you where to go.
C	I think you forget important thing: Economy and delivery on time. That is really important thing, even in Agile. You need to discuss issues such as: how far or how much you deliver so far in %, e.g., 5 or 60% of the product backlog? where are we now? and how much money this spends? those are always questions that customers will ask.

D	It is truly benefit to be able trace back to formal business stakeholders. Obviously, you check who is the stakeholder for this process? The larger the project the more stakeholder they are, the more risky when they want to change something without knowing who is stakeholder. Essentially, I would say for requirement, traceability would be like service subscription that business manager should subscribe, like you buy the insurance. Additionally, traceability helps to see the progress of the project.
E	Like I said, to me it is always to be able to have clients, you need to show what have done, why done, who added for this, why decision made, what impact it had not you escalated , etc. It comes down to commercial aspects of interacting with client basically. I am talking about those, who are responsible for handling clients, who have business relations with clients. They want stuffs all the time for sure, but when it comes to pay, they do not want to pay for all stuffs. You need to have your motivation clearly, you keep stuffs together.
F	I think the biggest argument is software maintenance. That is one affecting company developing software and customers. It is big problem not having tracing links, it is a big risk. I think it should be something for the organization, who arranges and makes time for this in the budget, e.g. you have to spend 02 hours documenting, I think developers will do. We write small things in one place, we try to keep track which person working on what project and what task that person has been working for the project, so you know who to ask. I see relevant points to do detailed documents, update links. But we have no resources to do those activities.
G	I think a lot of what you do when you add traceability like buying insurance. That means you want, e.g. if you do not handle errors, you will end up with errors. It make easier to make route causes analysis, to fix problem. It also is good when doing customers supports, when customers confirm situations with problems or questions, it make a lot easier to ... questions. It will give good integrity picture. If you have good traceability, you have a lot easier to give that feeling to customers, they will get have good system, you know what you are talking; valuable supports when customers have questions or problems. They are hard values. Soft value is you give the better impression on company.
H	It seems that all benefits you have discussed in the filled in form and I think most important to ensure customers satisfaction by doing traceability.
I	I think it is the way to improve the way of working and cooperation between team members. We have Scrum Board tool and traceability helps to improve communication between teams sitting different locations.
J	If you do not add traceability, you will be trouble sometime. Traceability will have all these things as you know something is broken down; you need to be able to trace. I agree with all purposes of traceability in HANDOUT-02. It makes easier to work when you know where to find information because all work related to information. Right? If we deliver products not working, we must fix this. Thus, everybody here concern about traceability. Traceability makes easier to explain how things have been done. If you have artifacts, they help you out to explain.
K	To keep the developments satisfy change of requirements because tasks are prioritized and implemented in short sprints. Since the implementations are changed at small steps, it's easier to maintain the code hence usually maximized the code stability. The implementation is done with simplicity in mind, so it's much easier for different developers to work on the same code. As the result, more stable releases are delivered to customers.
M	I think we reach our goals, we set goals, date for finished products. So traceability from estimated requirements, then going on to solutions phase, going on to functionality that satisfies requirements. I will use this matrix to keep track progress of the project. Traceability is supporting for tracking progress.

3. **Combining interviews and literature:**
Descriptions made by interviewees in the Table 7.18 above were interpreted, summarized and classified

according to 05 categories in the Table 7.17. Then, these descriptions were merged with findings of literature in the Table 7.17 to make the Table 7.19, on which the Analysis section was based:

Table 7.19: Synthesizing the findings from interviews and literature

Type benefits	Reported by company	Reported by literature
Checking progress	Companies C, D, E, I, M mentioned this benefit	this benefit was found in papers [79, 97, 90, 150].
Customer focus	it was expressed by companies B, C, G	Literature [97] mentioned this benefit.
Buying insurance	Companies D, G, H and J emphasized this benefit	
Saving resources	it was stated by companies A	This kind of benefits was mentioned in papers [79, 129, 84].
Cooperation & share knowledge	it was mentioned by companies I, A	Papers [23, 89, 63] mentioned this benefit.
Improve software quality	this benefit was mentioned by company F	it was found in papers [23, 79, 39, 51, 63, 90, 100].

4. **Analysis and discussion of this question**

4.1 **Checking progress:**

This type seemed to be the most common benefit according to findings of interviews and literature. Not only do we check the progress of the whole project, but we can also check the progress of each team member. Interviewees told that Agile companies often ask themselves: where are we now? are we 40%, 60% or 80% ready for delivery? It could be assumed that raising such questions like these may help the development team to quickly capture the whole picture of the on-going project. Besides, tracing links between a stakeholder with his own software artifacts seems to help to check the progress of an individual member. For instance: a tester writing a test report for a subsystem, could tell his manager: how many hours from this moment for him to write and finish this test report. Findings of literature also supported for those of interviews. In his related work, Jacobsson [79] stated that doing traceability helps to track private tasks, progress of team members. Furthermore, primary studies [90, 100] wrote that implementing traceability could measure the progress of the project.

4.2 **Customer focus:**

Companies C, H and E stated some reasons why customer focus was the important benefit. First, doing traceability could enable you to answer customers, who always raise questions related to delivery on time and economic aspects. Second, the focus needs to be put on the customer because if we do traceability well, it may be much easier to transfer comfortable and confident feelings to customers, making sure with customers that they will receive good software products. "Customer focus" criteria could help us to feel confidently for what we are talking about and show customers what have been done and why? who made this decision and why?. This was confirmed by the interviewee E, who shared his specific story when telling his customer "...I kept track of this as you said to me before ...". This finding was in line with insights found in [97], in which Christopher reported that traceability could be considered as a key enabler in maintaining the customer focus during the software development process.

4.3 **Buying insurance:**

More interestingly, the term "buying insurance" was mentioned by four companies D, G, H and J as the very important benefit. However, no literature in the Table 7.19 mentioned this important benefit.

This *inconsistency* could be explained that feedback of interviews were evaluated and provided by project leaders in actual projects. In contrast, findings of primary studies were not evaluated in the industry context and were mainly reported from small experiments. As we see that change occurs continuously throughout the development process, so there could be so much risk with late change, particularly for big and long-lived projects.

It could be assumed that if we do not implement traceability early and sufficient enough, the system may be ended up with errors or bugs, which may cost us much money. For example, an interviewee said "...if you caught these bugs in an unit test, we lost 20 minutes of fixing bugs in your code ... if you caught bugs in integration test, we lost 3 weeks for coordinating resources to fix for changes... and if we do not fix these bugs, the client may be lost 30 millions SEK of revenue ...". Therefore, doing traceability could be considered as if a software company buys an insurance for its software products.

4.4 **Cooperation and shared knowledge:**

Implementing traceability in Agile will help to tighten cooperation and spread knowledge between stakeholders as most interviewees mentioned. It could imply that since different roles need to participate in a software project, so all roles need to do traceability. For instance: business analysts, marketing people and software developers need to communicate to understand what will be changed to meet a new requirement from the customer. Companies B, I and K said that they do not depend much on traceability tools, but on the communication channels within the team: e.g., phone, face-to-face or Video meetings to discuss traceability activities. This was in line with findings in literature. Papers [13, 63, 40] described that doing traceability could help to increase understandings, cooperations or collaboration between stakeholders.

Strong cooperation also needs to be established between different levels of management, particularly from high managers of companies. For example: companies A, B and M stated that managers at high level need to give dedicated resources to support traceability tasks, to motivate and explain with team members about the importance of traceability. In addition, doing traceability will help to spread knowledge within the team as mentioned by companies I and B. This feedback was supported by literature, which stated that traceability may help to provide a way for organizational learning [23] or spread knowledge within an organization [89]. This implies that employees participating in doing traceability, may create the environment of learning, in which members will learn and help each other. For instance: an experienced member will help a new one how to become familiar and use the traceability tools.

4.5 **Improve software quality:**

Primary papers [23, 79, 39, 51, 90, 63, 100] mentioned specific benefits grouped under the category: "to improve software quality" in the Table 7.19. However, this is *inconsistent* with findings in the Table 7.19 where only company F mentioned this type of benefits. The reason could be that the concept of "improve software quality" may be too abstract that practitioners may not know what this concept may actually refer to as mentioned by many companies. For instance: company B from United States said that "improve software quality" is too general, covering many things and assumed that few benefits could be gained via doing traceability activities. Compared to other 11 Swedish companies, this USA company seems to ignore formal approaches or frameworks to conduct traceability. It could be deemed that this company does not sell a finished software system, but makes software tools to support for their main products: semi-conductor or chip. Its software projects may, therefore, be very simple and just need few documents. Though it was not mentioned directly, it could be said that this USA company conducted traceability tasks to some extent, which could help to improve software quality.

4.6 **Save resources:**

This type of benefits was mentioned in papers [79, 97, 129, 40, 84]. However, this seems to be *inconsistent* with descriptions of practitioners in the Table 7.18 where only 01 company slightly mentioned to this benefit. This *inconsistency* could be explained that in literature, researchers compared resources spent on one experiment in two situations: with and without traceability and found that doing traceability might have saved resources. In reality, doing traceability requires companies to spend more resources as mentioned by companies at "Cost effectiveness" in the RQ4. "Save resource" may probably be understood in the sense that if we do not conduct traceability early at sufficient levels, what will happen to a software system?. Thus, it could be concluded that "save resource" should not be regarded as the major benefit.

Chapter 8

Validity Threats

Several types of threats to validity were discovered in this study. Most of them could be associated with the implementation of the industry interview. According to Wohlin [155], validity threats can normally be categorized in conclusion, internal, construct and external threats. Sequential sections below will describe specific threats facing this thesis:

8.1 Internal validity

Internal validity inspects whether or not accurate deductions are derived from gathered data [156]. The threats associated with internal validity can be reduced by making the referral to multiple perspectives on the topic of interests.

8.1.1 Threats of conducting interviews

Threats related to interviews in this paper could be that subjective questions can lead to a poorly designed. Interview questions of this paper were strictly designed on the basis of the problems and issues, which were relevant to literature, but the emphasis was put on the industry. The threat of missing any important questions related to what important type of traceability, what traceability mechanisms and if these mechanisms worked well in Agile, were minimized by careful reviews of thesis supervisor and interviewees were asked to add up if they thought of missing any thing during interviews.

Throughout implementing interview process, some shortcomings or threats actually occurred. This thesis author lacked skills to ask questions, used motivational probes. The traceability Matrix was rather complex that 50% of interviewees asked to delay and supplemented later on. However, it cost so much time to get the supplement since most interviewees are holding senior position and so busy. Thus, some supplements were provided with less care and insufficient information. In addition, though interviewees were confused with the traceability Matrix, some of them still used terms/concepts in this Matrix to describe types of traceability. Thus, there seemed to be bias information or leading answers provided by interviewees since some interviewees seemed to be forced to follow the format of HANDOUT-03 when providing answers to RQ1. Furthermore, lacking skills to ask questions in proper ways in this paper resulted in some irrelevant information.

8.1.2 Threats of reviewing papers

Publication bias refers to the general problem that positive research outcomes are more likely to be published than negative ones [157]. The objective nature of classification was done according to the mutual understanding of the researcher. Its subjective nature, therefore, could not be ruled out which can be termed as the threat to this study. The threat, with which needed to be coped in the literature review is associated with publication biasness. An attempt was made to strictly follow steps to identify literature in quality and well known databases, to summarize studies and extract data from primary papers. By doing so, this thesis author has tried to minimize the internal validity threat.

8.2 External validity

8.2.1 Threats to the industry interview

At the beginning, the formal cost and benefit analysis of the research topic was sent to solid knowledge & experienced experts to recommend for the interview participation. The contacted people, who really interested in this research topic, replied and showed his interest. It could be said that there was no threat that interviewees lacked of interests at the beginning. However, during actual interviews, a couples of interviewed subjects expressed less interests in particular questions or in their supplements after the interviews. The thesis author tried to mitigate these issues by providing more detailed and proper explanations.

8.2.2 Threats to the literature review

Looking all primary studies of this thesis, it is recognized that these reviewed papers were all published since 2003 up to date. It could be deemed that a relevant study discussing this research topic had been published before 2003 or during this period time. To minimize this threat, this thesis author tried to follow instructions to formulate searched strings and techniques to identify relevant papers. For example, Snowball sampling search was used.

8.3 Construct validity

Construct validity valuates the utilization of accurate definition and measures associated with the variables according to Creswell [156]. The construct, e.g., the search strings utilized for the literature review, was well defined and suggested by the thesis supervisor. So it could rule out threats related to the constructs.

For the industry interview, the data was collected in the form of tacit knowledge. Some risk of misinterpretation could occur when trying to make knowledge more clear because tacit knowledge could be interpreted in different ways. This kind of risk was mitigated by relating, checking and comparing between relevant data available in literature with explicit interpretations of collected data.

8.4 Conclusion validity

According to Wohlin [155], conclusion validity known as reliability, is associated with the research study results and it assures that reported results will lead to correct and reliable conclusions.

8.4.1 Threats to the industry interview

The language fluency is very essential during interviews. To reduce this risk, the interview guide was carefully and clearly prepared in advance. Also, interviewees had some time to review questions during off-line interviews and parts of time-consumed question were sent to remote companies before the official interview. However, since English is not the native language of the thesis author, so misunderstandings did occur several times. In addition, a couple of interviewees also misunderstood a couple of times during their interviews as English is not their mother tongue either.

8.4.2 Threats to the literature review

Before starting the literature review process, the preliminary search was conducted by using some formulated search phrases with two search engine, i.e. Google Scholar and Summon (mentioned in previous chapters). The result of searching literature, in reality, was quite low since the research topic of this thesis could be deemed as the very new topic. To cope with this, Snowball sampling search was supplemented in order to find additional important papers.

Chapter 9

Conclusion

This section is to summarize the work done in this thesis. First, some main contributions are illustrated. Next, each research question will be revisited to summarize major conclusions. At the end, some suggestions for future work will be provided.

In overall, the main contributions of this thesis are:

- Providing the comprehensive overview of the state-of-art in software traceability and Agile by creating two corresponding maps, on which readers are provided the knowledge area.
- Identifying the important types of traceability in Agile software projects reported from both the industry and literature.
- Supplementing findings of challenges against benefits when implementing traceability in Agile projects reported from the industry and literature.
- Identifying what traceability mechanisms are used, and then these mechanisms will be commented and evaluated by practitioners.

The goal of this study has two main folds. On the one side, this paper is deemed to provide the basic body of knowledge for software traceability. On the other side, this study provides quite details to investigate how software traceability is implemented in an Agile project according to findings of the industry and literature. The following sections will go into each specific research question.

9.1 Research questions revisited

RQ1:What types of traceability are important for software development in organizations subscribing to being Agile?

To begin with, literature were reviewed to identify major types of software traceability. After that, these types were described and evaluated by practitioners. In this thesis, 08 types of traceability: Dependency, Evolution, Rationale, Contribute, Refinement, Satisfy, Overlap and Conflict were discussed. These types were discussed and considered as the important types in traditional software development process. In Agile, however, only Rationale and Contribute were considered as the most important types of traceability since stakeholders of a software system always need to pay attention to the reason why something is done and by whom. Refinement was considered as the third important type since refining an artifact or element of a system may be the very often activity during the development life-cycle of a system. However, 05 remaining types were not regarded as the important. Specifically: Dependency, Evolution and Satisfy were less important since the team development members assume that these three types seem to be obvious and they know how to use many available approaches to track them quite easily. Finally, Conflict and Overlap were not mentioned at all since tracking these two types of traceability links could be considered as the pre-activity before doing traceability. In other words, Conflict and Overlap need to be inspected, analyzed and removed before conducting traceability.

RQ2: What mechanisms are used by Agile companies to achieve traceability?

A mechanism is collection words, which refers to: tools, methods, approaches, frameworks or techniques to help practitioners to achieve traceability. It could be said that there may be many mechanisms available and selecting a traceability tool to use may depend on many criterion such as: the type, size or length of a project. It was found that all companies needed the combination of manual and automated tools in order to achieve traceability. On the one side, companies used important traditional approaches such as: MS Office, Backlog, while board, communication channels/ways. On the other side, companies used software tools to conduct traceability. Small companies mainly used open sources or self-developed software tools. In contrast, big companies bought expensive commercial software tools such as: Team foundation server, IBM rationale, Visio Studio and JIRA. It was also found that most companies may need to combine software development processes to make full use of resources, e.g. global companies re-used traceability tools that were initially designed to support traditional software development methods.

RQ3: What mechanisms work well in Agile development organizations?

Companies used different mechanisms to achieve traceability as mentioned in the RQ2 since there was no default tool for all projects. Global or big companies stated that commercial tools bought from providers such as IBM and Microsoft did not work very well in their organizations because these tools seemed to be complicated to use, did not meet actual needs of each specific project and some tools were mandatory for all projects that membered companies within the global organization had to follow. On the contrary, medium and small companies exploited and used self-developed or open source tools quite effectively since their development teams made analysis, evaluated the level and need of traceability for each project before selecting what tools to use. In addition, it was found that knowledge and experience of employees would be the decisive factors to get traceability mechanisms to work effectively. For example: if a company has experienced team members to use Test-Driven Development (TDD), this practice was strongly suggested by both interviews and literature to be the best tool to achieve traceability in Agile software development.

RQ4: What challenges exist for realizing traceability in Agile organizations?

Firstly, findings of literature and descriptions of practitioners were classified into 05 categories related to: Cost effectiveness; Knowledge & experience; Cooperation, regulations or organization; Technology or technical and other difficulties. Among these 05 types of difficulties, knowledge and experience of team members were reported to be the most challenging because these factors may be essential to implement traceability and get mechanisms work effectively. Another difficulty was found in association with cooperation of different units within an organization. Besides, the commitment of team members was also considered as the challenge because if employees of a company do not cooperate, keep commitments and collaborate with each other, it will be impossible to conduct traceability. In addition, policies/regulations, committed resources given to traceability activities seemed to be challenging since these factors could motivate, encourage and explain the right goals of traceability tasks to the team members.

RQ5: What benefits exist in realizing traceability in Agile organizations?

Similarly to the RQ4, findings in papers and feedbacks of practitioners were categorized into some main types of benefits. First of all, doing traceability was seen as if “buying insurance” and this could be the most important benefit since a software company may face serious troubles later on if the company does not “buy insurance” or conduct traceability. Another important benefit was “customer focus” since the development team need to answer questions and provide comfortable, confident feelings to the customer. In addition, conducting software traceability was found to surely track the progress of the whole project since traceability activities help to check the progress and tasks of individual members very often. Furthermore, implementing traceability was reported to spread knowledge and tighten cooperation in an organization since employees need to cooperate, communicate and help each other when doing traceability.

9.2 Future work

The results of this study could be considered as the foundation for further investigation in the area of software traceability in Agile software development. The thesis author is planning to continue on one of several directions for his further work as followed:

Creating and experimenting a traceability framework in an Agile project

Basing on this thesis results, a framework specified for Agile software development will be proposed. Then, this framework will be experimented and validated in the industry.

Conducting in-depth case studies about traceability mechanisms

Though insight, feedbacks of practitioners in this thesis are important, they are very general. Thus, conducting some in-depth case studies about traceability mechanisms in Agile companies may be indeed necessary. Perfectly, case studies should be conducted in different size and type of organizations or companies to get overview of existing mechanisms. Special focus may be put on investigating and analyzing how traceability mechanisms could help companies to achieve main philosophies of Agile.

Identifying a model to evaluate and measure the effectiveness of traceability mechanisms

Evaluating the effectiveness of traceability mechanisms could be very important since companies spend large amount of resources on using mechanisms to achieve traceability. First, the initial model will be developed to get evaluations from experts both from the academia and industry. After that the model will be experimented and evaluated in the industry.

Chapter 10

Reference

[1] I. Alexander, “Towards Automatic Traceability in Industrial Practice”, Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2002), Edinburgh, UK, September 2002.
[2] I. Alexander, “Semi-Automatic Tracing of Requirement Versions to Use Cases - Experience and Challenges”, Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003), Canada, October 2003
[3] R. Akbar et al., 2010, ”Client’s Perspective: Realization as a New Generation Process for Software Project Development and Management”, IEEE Computer Society, pg: 191-195
[4] A. Agbulos and SM. AboutRizk, 2003, ”An application of lean concepts and simulation for drainage operations maintenance crews”, University of Alberta, Canada
[5] G. Antoniol et al., “Recovering Traceability Links between Code and Documentation”, IEEE Transactions on Software Engineering, 28(10), 970-983, October 2002
[6] HU. Asuncion et al., 2010, “Software traceability with topic modeling” Proceedings of the 32nd ACM-IEEE International Conference on Software Engineering - Volume 1
[7] HU. Asuncion and RN. Taylor, 2012, “Automated Techniques for Capturing Custom Traceability Links Across Heterogeneous Artifacts”, Computer Science, Part 2, pg:129-146
[8] A. Aurum and C. Wohlin, “Engineering and Managing Software Requirements”, Springer, Berlin Heidelberg, 2005
[8] W. Apel et al., 2007, ””Value Stream Mapping for Lean Manufacturing Implementation””, Huazhong University of Science Technology”
[9] G. Aiello et al, 2007, ”An Agile methodology for Manufacturing Control Systems development”, Industrial Informatics, 5th IEEE International Conference, pg: 817-822
[10] T. Aftab, ”Improved software quality with Agile processes”, www.ephlux.com
[11] M. Asta and A. Vaidas, 2008, ”Bottlenecks in Agile Software Development Identified Using Theory of Constraints (TOC) Principles”, Master thesis, IT University of Goteborg
[12] B. Appleton et al, “Lean Traceability: a smarttering of strategies and solutions”, Configuration Management Journal 2007
[13] KM. Anderson et al., ”Towards large-scale information integration”, Proceedings of the International Conference on Software Engineering (2002)
[14] J. Bayer and T. Widen, “Introducing Traceability to Product Lines”, Proceedings of the Software Product Family Engineering: 4th International Workshop, PFE 2002, Bilbao, Spain, Lecture Notes in Computer Science, Springer-Verlag, ISSN:0302-9743.
[15] M.F. Bashir, M.A. Qadir, “Traceability Techniques: A Critical Study”, IEEE Multitopic Conference (INMIC’06), IEEE, 2006, pp. 265-268.
[16] U. Bellur and Vallieswaran, 2006, “Versioning Scheme for Consistent Evolution of OO Applications”,

XIII Asia Pacific Software Engineering Conference, IEEE
[17] B. Boehm, 2002, ”Get Ready For The Agile Methods With Care” IEEE Computer Society, vol. 35, pp. 64-69
[18] K. Beck, 1999 ”Embracing Change With Extreme Programming,” IEEE Computer Society, vol. 32, pp. 70-77
[19] K. Beck, ”Extreme Programming Explained: Embrace Chage”, second ed., Addison-Wesley, 2004, ISBN 978-0321278654.
[20] J. Cleland-Huang, C.K. Chang, and J.C. Wise, “Automating performance-related impact analysis through event based traceability”, Journal of Requirements Engineering, vol. 8, no. 3, Springer, 2003, pp. 171-182.
[21] J. Cleland-Huang, 2006, “Just Enough Requirements Traceability”, Proceedings of the 30th Annual International Computer Software and Application Conference, IEEE
[22] J. Cleland-Huang, 2012, “Traceability in Agile Projects”, Computer Science, part4, pg: 265-275
[23] J. Cleland-Huang et al, 2012, “Software and Systems Traceability”, Springer
[24] J. Cleland-Huang and D. Schmelzer., “Dynamic Tracing Non-Functional Requirements through Design patter Invariants”, Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003), Canada, October, 2003
[25] J. Cleland-Huang et al., “Automating Speculative Queries through Event-based Requirements Traceability”, Proceedings of the IEEE Joint International Requirements Engineering Conference, Essen, Germany, September 2002
[26] J. Cleland-Huang, Requirements Traceability When and how does it Deliver more than it Costs?, 14th IEEE International Requirements Engineering Conference (RE’06), IEEE, 2006, pp. 323-323.
[27] J. Cleland-Huang et al., Event-based traceability for managing evolutionary change, IEEE Trans. on Software Engineering 29(9) (2003) 796810.
[28] J. Cleland-Huang and C. K. Chang, Supporting event based traceability through high-level recognition of change events, in IEEE Proc. Intl Computer Software and Applications Conf. (COMPSAC), 2002.
[29] P. Constantopoulos et al., “The Software Information Base: A Server for Reuse”, VLDB Journal, 4(1), 1-43, 1995
[30] G. Cysneiros et al., “A Traceability Approach for i* and UML Models”, Proceedings of 2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems -ICSE 2003, May 2003
[31] F. Cafer and S. Misra, 2009, ”Effective Project Leadership in Computer Science and Engineering”, Computational Science and its Applications, pg:59-69
[32] J. Chao, 2005, ”Balancing hands-on and research activities: a graduate level agile software development course”, IEEE Xplore, pg:306-311
[33] A. Cockburn, 2006, ”Agile Software Development: The Cooperative Game”, Addison Wesley, Second Edition
[34] A. Cockburn, ”Crystal Clear: A Human-Powered Methodology for Small Teams”, Addison-Wesley, 2004, ISBN 0-201-69947-8.
[35] K. Conboy and B. Fitzgerald, 2004, ”Toward a conceptual framework of agile methods: a study of agility in different disciplines”, Proceedings of XP/Agile Universe, Springer Verlag
[36] L. Cao and B. Ramesh, 2008, “Agile Requirements Engineering Practices: An Empirical Study”, IEEE Computer Society, vol.25, issue.01, pg: 60-67
[37] L. Cao et al, 2009, A framework for adapting agile development methodologies, European Journal of Information Systems, pg: 332-343
[38] L. Cao and B. Ramesh, 2008, “Agile Requirements Engineering Practices: An Empirical Study”, IEEE Computer Society, vol.25, issue.01, pg: 60-67
[39] J.W. Creswell, Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, Sage publications, 2002.
[40] A.M. Davis, “The analysis and specification of systems and software requirements”, In Systems and Software Requirements Engineering. IEEE Computer Society Press, 1990, 119-144
[41] M. Deng et al, 2005, “Retrieval By Construction: A Traceability technique to support verification and

validation of UML Formalization", International Journal of Software Engineering and Knowledge Engineering, Vol.15, No.5, pg:837-872
[42] J. Diaz et al., 2011, "Change Impact Analysis in Product-Line Architectures", Computer Science, Springer-Verlag Berlin Heidelberg, pg:114-129
[43] J. Dick, "Rich Traceability", Proceedings of the 1st International Workshop on Traceability for Emerging forms of Software Engineering (TEFSE-02), Edinburgh, UK, September 2002
[44] T. Dyba and T. Dingsoyr, 2009, "What Do We Know about Agile Software Development?", IEEE, vol.26, Issue.5, pg: 6-9
[45] RA. Dunningt and TM. Chichert, 2001, "Applying Lessons from Lean Production Theory to Transit Planning", Paper at ASCE 8th International
[46] T. Dyba et al, 2007, "Are two heads better than one? On the effectiveness of pair-programming", IEEE Software vol.24, issue. 6, pg: 1013.
[47] R.M. Davison, 2005, On peer review standards for the information systems literature. Communications of the Association for Information Systems, 16(4), 967-980.
[48] L. Delgadillo and O. Gotel, 2007, Story-Wall: A Concept for Lightweight Requirements Management, 15th IEEE International Requirement Engineering Conference, pg:377-378
[48] A. Espinoza and J. Garbajosa, 2011, "A study to support agile methods more effectively through traceability", Innovation Systems Software Engineering, pg:53-69.
[49] V. Esther Jyothi and N. Rao, 2011, "Effective Implementation of Agile Practices", International Journal of Advanced Computer Science and Applications, vol.2, no.3, pg:41-48
[50] M. Edwards and SL. Howell, 1991, "A Methodology for System Requirements Specification and Traceability for Large Real-Time Complex System", technical report, UU. Naval Surface Warfare Center
[51] A. Egyed, "A Scenario-Driven Approach to Trace Dependency Analysis", IEEE Transactions on Software Engineering, Vol. 9, No. 2, February, 2003
[52] A. Egyed, P. Gruenbacher, "Automatic Requirements Traceability: Beyond the Record and Replay paradigm", Proceedings of the 17th IEEE International Conference on Automated Software Engineering (ASE), Edinburgh, UK, September, 2002 Whiley Online Library, Vol.12, Issue.4, pg:143-157
[53] R. Feldt http://www.cse.chalmers.se/ feldt/advice/
[54] O. Gotel and A. Finkelstein, "Contribution Structures", Proceedings of 2nd International Symposium on Requirements Engineering, (RE '95), 100-107, 1995.
[55] O. Gotel et al, 2012, "Traceability Fundamentals", Computer Science, SpringerLink, part 1, pg:3-22.
[56] O. Gotel and A. Finkelstein, 1994," An analysis of the requirements traceability problem", Requirements Engineering of Proceedings of the First International Conference, IEEE, pg: 94-101.
[57] A. Ghazarian, 2010, "Reliability in Agile Software Engineering: A Dilemma", Annual Technical Report, IEEE
[58] A. Ghazarian, 2008, "Traceability Patterns: An Approach to Requirement-Component Traceability in Agile Software Development", Proceedings of the 8th WSEAS International Conference on APPLIED COMPUTER SCIENCE, pg: 236-241
[59] T. Gorschek, "Requirements Engineering Supporting Technical Product Management", PhD Thesis no. 2006:01, ISBN 91-7295-081-1, Blekinge Institute of Technology, Ronneby, Sweden.
[60] Y. Ghanam and F. Maurer, Linking Feature Models to Code Artifacts Using Executable Acceptance Tests, Computer Science, SpringerLink, Vol.6284/2010, pg:211-225
[61] A. Gomes and A. Pettersson, Market-Driven Requirements Engineering Process Model MDREPM, Masters Thesis no. MSE-2007-06, Blekinge Institute of Technology, Ronneby.
[62] Z. Hussain et al., 2009, "Current State of Agile User-Centered Design: A survey", Springer-Verlage Berlin Heidelberg, vol.5889/2009, pg: 416-427
[63] J.H. Hayes, A. Dekhtyar, J. Osborne, "Improving Requirements Tracing via Information Retrieval", Proceedings of the 11th IEEE International Requirements Engineering Conference, Monterey Bay, 2003
[64] J.H. Hayes et al, 2009, "Towards Traceable Test-Driven Development", IEEE workshop, pg:26-30
[65] F. Hamzeh and E. Bergstrom, 2010, "The Lean Transformation: A Framework for Successful Implementation of the Last Planner TM System in Construction", Colorado State University

[66] J. Highsmith and A. Cockburn, 2001, "Agile Software Development: The Business of Innovation," IEEE Computer Society, vol. 34, pp. 120-122
[67] K. Hiranabe: Kanban applied to software development: From agile to lean (2008), http://www.infoq.com/articles/hira
lean-agile-kanban
[68] M. Hennink et al., (2011). Qualitative research methods, 1st Edition, SAGE Publications
[69] SE. Hove and B. Anda, 2005, "Experiences from Conducting Semi-Structured Interviews in Empirical Software Engineering Research", Software Metrics, 2005. 11th IEEE International Symposium, pg:10-23
[70] A. Hart, 1999, "Doing a Literature Review: Releasing the Social Science Research Imagination", SAGE Publications Ltd, p.13
[71] E. Hossain et al, Using Agile Practices in Global Software Development: A Systematic Review", IEEE Computer Society, pg: 175-184
[72] S. Imtiaz and N.Ikram., 2008, "Impact Analysis from Multiple Perspectives: Evalution of Traceability Techniques", Software Engineering Advances IEEE ICSEA 08, The Third International Conference pg:456-464
[73] M. Ikonen et al, 2010, "Exploring the Sources of Waste in Kanban Software Development Projects", IEEE Computer Society, pg: 376-381
[74] M. Ikonen et al, 2011, "On the Impact of Kanban on Software Project Work An Empirical Case Study Investigation", IEEE Computer Society, pg:305-314
[75] A. Jackson et al, 2006, "Towards Traceability between AO Architecture and AO Design", Early Workshop, pg:1-8
[79] M. Jacobsson, 2009, "Implementing Traceability In Agile Software Development", Department of Computer Science, Lund University, Sweden
[80] M. Jarke, 1998, "Requirements Tracing", Comm. ACM, vol.41, no.12, pg:32-36
[81] M. Javed et al, 2010, "Mapping The Best Practices of XP and Project Management: Well defined approach for Project Manager", Journal of Computing, vol.2, issue.3, pg:103-108
[82] VE. Jyothi et al, 2012, "Effective Implementation of Agile Practices-Incoordination with Lean Kanban", International Journal on Computer Science and Engineering, vol.4, No.5, pg:87-92
[83] S. Jalali and C.Wohlin, 2010, "Agile practices in global software engineering-A systematic map", International Conference on Global Software Engineering (ICGSE), Princeton NJ
[84] VE. Jyothi and KN. Rao, 2011, "Effective Implementation of Agile Practices", International Journal of Advanced Computer Science and Applications, Vol.2, No3
[85] N. Kececi, J. Garbajosa, and P. Bourque, "Modeling Functional Requirements to Support Traceability Analysis", IEEE International Symposium on Industrial Electronics, vol. 4, IEEE, 2006, pp. 3305-3310.
[86] H. Kaindl, "The Missing Link in Requirements Engineering", Software Engineering Notes, June 1992, pp. 498-510
[87] V. Knethen et al., "Systematic Requirements Recycling through Abstraction and Traceability", Proceedings of the IEEE International Requirements Engineering Conference, Germany, September 2002.
[88] M. Kunz et al, 2008, " How to Measure Agile Software Development", Software Proes and Product Measurement, vol.4895, pg:95-101
[89] V. Kirova et al, 2008, "Effective Requirements Traceability: Models, Tools, and Practices",
[90] L. Klimpke and T. Hildenbrand, 2009, "Towards End-to-End Traceability", IEEE Computer Society, Software Engineering Advances, pg: 465-470
[91] A. Kozlenkov and A. Zisman, "Are their Design Specifications Consistent with our Requirements?", Proceedings of IEEE Joint International Requirements Engineering Conference - RE'02, Essen, September 2002.
[92] B.A. Kitchenham and Charters, S. (2007) Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3, Keele University, EBSE Technical Report, EBSE-2007-01.
[93] C. Larman, "Agile and Iterative Development: A Managers Guide", Boston: Addison-Wesley (2004)
[94] Y. Levy and T.J. Ellis, "A systems approach to conduct an effective literature review in support of information systems research," Informing Science: International Journal of an Emerging Transdiscipline, vol. 9, 2006, pp. 181-212.

[95] P. Letelier et al., 2005, "Customizing traceability in a software development process", Information System Development, pg:137-148
[96] P. Letelier, 2002, "A Framework for Requirements Traceability in UML-based Projects", Proceedings of the 1st International Workshop on Traceability for Emerging Forms of Software Engineering (TEFSE'02), Edinburgh, UK, September 2002
[97] C. Lee et al., 2003, "An Agile Approach to Capturing Requirements and Traceability", International Workshop on Traceability
[98] G. Lee and W. Xia, 2010, "Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development", ACM Digital Library, vol.34, issue.01
[99] P.D. Leedy and J.E. Ormrod, 2005, Practical research: Planning and design (8th ed.). Upper Saddle River, NJ: Prentice Hall.
[100] AD. Lucia and A. Qusef, 2010, "Requirements Engineering in Agile Software Development", Journal of Emerging Technologies in Web Intelligence, Vol.2, No.3
[101] J. I. Maletic et al., An XML based approach to support the evolution of model-to-model traceability links, in Proceedings of 4th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE05), 2005
[102] P. Mder and J. Cleland-Huang, 2010, "A Visual Traceability Modeling Language", Model Driven Engineering Languages and Systems, pg: 226-240
[103] A. Marcus and J.I. Maletic, "Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing", Proceedings of 25th International Conference Software Engineering, 2003
[104] K. Mohan and B. Ramesh, "Managing variability with Traceability in product and Service Families", Proceedings of the 35th Hawaii International Conference on System Sciences, IEEE, 2002
[105] J.I . Maletic et al., "Using a Hypertext Model for Traceability Link Conformance Analysis", Proceedings of the 2nd International Workshop on Traceability for Emerging Forms of Software Engineering (TEFSE'03), Canada, October 2003
[106] J. McAvoy and D. Sammon, 2009, "Agile Methodology Adoption", University College Cork, Ireland
[107] A. Mahfouz et al., 2011, "Integrating Current State and Future State Value Stream Mapping with Discrete Event Simulation: A Lean Distribution Case Study", IARIA, pg: 161-168
[108] J. McAvoy and T. Butler, 2009, "A Failure to Learn in a Software Development Team: The Unsuccessful Introduction of anAgile Method", Information Systems Development, pg:1-13
[109] A. Misharand and S. Misra, 2010, "People Management in Software Industry: The Key to Success", ACM Digital Library, vol.35, No.06, pg:1-4
[110] A. Mostafaeipour and MS. Fallahnezhad, 2010, "Implementation of Agile Manufacturing into Value Engineering Technique for Industries", Proceedings of IDMME, pg:1-8
[111] R. McCauley,2001, "Agile Development Methods Poised to Upset Status Quo," SIGCSE Bulletin, vol. 33, pp.14 - 15
[112] MM. Mller and WF. Tichy, 2001, "Case Study: Extreme Programming in a University Environment," presented at 23rd International Conference on Software Engineering, Toronto
[113] P. Maher, 2009, "Weaving agile software development techniques into a traditional computer science curriculum" LasVegas NV April, pp: 16871688
[114] J. Martinez at al, 2009, "Software Product Line Engineering Approache for Enhancing Agile Methodologies", Computer Science, vol.31, part.5, pg:247-248
[115] M. McHugh et al., Barriers to Adopting Agile Practices When Developing Medical Device Software, Springer-Erlag Berlin Heidelberg 2012, p.141-147
[116] V. Oza et al, 2011, "Attaining High-performing Software Teams with Agile and Lean Practices: AnEmpirical Case Study", University of Hensiki, pg: 1-4
[117] JD. Palmer, 1997, "Traceability", Software Requirements Engineering, R.H. Thayer and M. Dorfman, eds, pg:364-374
[118] P. Protsyk and K. Zhereb, "A Criteria-Based Approach to Classifying Traceability Solutions", the Proceedings of IEEE International EAST-WEST, pg: 1-7
[119] K. Pohl, "PRO-ART: Enabling Requirements Pre-Traceability", Proceedings of the 2nd IEEE

[120] K. Pohl, “Process-Centered Requirements Engineering”, John Wiley Research Science Press, 1996
[121] F. Pinheiro, “Formal and Informal Aspects of Requirements Tracing”, Proceedings of 3rd Workshop on Requirements Engineer), Rio de Janeiro, Brazil, 2000
[122] F. Paetsch et al, 2003, ”Requirements Engineering and Agile Software Development”, IEEE Computer Society, pg:308-313
[123] CSA. Peixoto and AEA. Estela, 2009, ”A Conceptual Knowledge Base Representation for Agile Design of Human-Computer Interface”, IEEE Computer Society, pg: 156-160
[124] M. Poppendieck and T. Poppendieck, ”Lean Software Development -An Agile Toolkit for Software Development Managers”, Addison-Wesley, Boston, 2003, ISBN 0-321-15078-3.
[125] F. Paetsch et al., 2003,Requirements engineering and agile software developement, WET ICE 2003, Proceedings. Twelfth IEEE, p.308-313
[126] S.R. Palmer and J.M. Felsing, A Practical Guide to Feature-driven Development, Prentice Hall, Upper Saddle River, NJ, 2002, ISBN 0-13-067615-2.
[127] F. Pinheiro and J. Goguen, ”An Object-Oriented Tool for Tracing Requirements”, IEEE Software, 52-64, March 1996.
[128] A. Qusef et al, 2010, “Recovering Traceability Links between Unit Tests and Classes Under Test: An Improved Method, Software Maintenance (ICSM)”, IEEE International Conference, pg: 1-10
[129] S. Ratanotayanon et al., 2009, “Supporting Program Comprehension in Agile with Links to User Stories”, IEEE, pg: 26-32
[130] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards, “Requirements traceability: Theory and practice”, Annals of Software Engineering, vol. 3, no. 0, Springer, 1997, pp. 397-415.
[131] B. Ramesh et al., “Implementing requirements traceability: A case study” Proceedings of the International Symposium on Requirements Engineering (1995)
[132] B. Ramesh, “Factors Influencing Requirements Traceability Practice”, Communication of the ACM, vol. 41, no.12, 1998, pp 37-44
[133] B. Ramesh and B. Jarke, “Towards Reference Models for Requirements Traceability”, IEEE Transactions in Software Engineering, 27(1), 58-93, 2001.
[134] R. Ramesh and V. Dhar, ”Supporting Systems Development Using Knowledge Captured During Requirements Engineering”, IEEE Transactions in Software Engineering, June 1992, 498-510, 1992.
[135] J. Reck et al, 2004, ”Improving Software Quality through Refactoring by means, Fraunhofer Institute for Experimental Software Engineering 67661 Kaiserslautern, Germany of Didactical Augmented Experience”,
[136] C. Robson, 2011, ”Real World Research”, 3rd Edition, Wiley
[137] UA. Raja and K. Kamran, 2008, ”Framework for Requirements Traceability- TLFRT supporting pre-RS post-RS traceability”, Master thesis, School of Engineering Blekinge Institute of Technology
[138] J. Richardson and J. Green, ”Automating traceability for generated software artifacts”, Proceedings of the International Conference on Automated Software Engineering (2004)
[139] K. Schwaber and M. Beedle, ”Agile Software Development with Scrum”, Prentice Hall, Upper Saddle River, 2001.
[140] V. Shukla et al, 2011, “A Graph-Based Requirement Traceability Maintenance Model”, ICSEA 2011: The Sixth International Conference on Software Engineering Advance, pg:161-165.
[141] J.M. Sharp et al, 1999, ”Working towards agile manufacturing in the UK industry”, International Journal of Production Economics, vol.62, issue.1-2, pg:155-169
[142] P. Sanchez et al, 2011, “A framework for developing home automation systems: From requirements to code”, The Journal of System and Software, vol.84, issue.6, pg:1008-1021
[143] G. Spanoudakis and A. Zisman, 2005, “Software Traceability: A Roadmap”, Software Engingeering Group, City University of UK
[144] G. Spanoudakis et al., “Rule-Based Generation of Requirements Traceability Relations”, Journal of Systems and Software, 72(2), pp. 105-127, 2004
[145] S.A. Sherba et al., “A Framework for mapping Traceability Relationships”, Proceedings of the 2nd International Workshop on Traceability for Emerging forms of Software Engineering (TEFSE 2003), Montreal,

Canada, Septeming (III WERber, 2003
[146] JG. Schneider and R. Vasa, 2006, "Agile practices in software development-experiences from student projects", IEEE Computer Society
[147] K. Schwaber and M. Beedle, Agile Software Development with Scrum, Prentice Hall, Upper Saddle River, 2001.
[148] G. Spanoudakis, Plausible and adaptive requirement traceability structures, in Proc.14th Intl Conf. Software Eng. and Knowledge Eng., 2002.
[149] G. Spanoudakis et al., "Revising Rules to Capture Requirements Traceability Relations: A Machine Learning Approach", Proceedings of the 15th International Conference in Software Engineering and Knowledge Engineering (SEKE 2003), San Francisco, USA, July 2003.
[150] P. Salipante et al., "A Matrix Approach to Literature Reviews," in Research in OrganizationaBl ehavior,B . M.S taw and L. L. Cummings(eds.), JAIP ress, Greenwich,C T, 1982, pp. 321-348.
[151] D. Tore and Torgeir, 2008, "Empirical studies of agile software development: A systematic review", Information and Software Technology, vol.50, issue.9-10, pg: 833-859
[152] M. VanHilst et al., 2005, "Repository Mining and Six Sigma for Process Improvement", ACM Digital Library, vol.30, issue.04, pg:1-4
[153] W.P. Vogt, 1999, Dictionary of Statistics and Methodology: A Nontechnical Guide for the Social Sciences, London: Sage.
[154] X. Wang, 2011, "The Combination of Agile and Lean in Software Development: An Experience Report Analysis", IEEE Computer Science, pg:1-9
[155] J. Webster and RT. Watson, 2002, "Analyzing the Past to Prepare for the Future: Writing a Literature Review", Management Information Systems Research Center, University of Minnesota, vol26, No.2
[156] S. Winkler and J. Pilgrim, 2010, "A survey of traceability in requirements engineering and model-driven development", Software System and Modeling, vol.9, num.4, pg:529-565
[157] L. Williams, A. Cockburn, 2003, "Agile software development: its about feedback and change", IEEE Computer Society 36 (6) pg: 3943.
[158] C. Wohlin, 2000, Experimentation in software engineering: an introduction, Kluwer Academic Publishers.
[159] P. Xu and B. Ramesh, "Supporting Workflow management Systems with Traceability", Proceedings of the 35th Hawaii International Conference on System Sciences, IEEE, 2002.
[160] U. Yllescas et al., "Trying to Link Traceability Elements in a General Agile Models Life Cycle", Science and Information Technology Program UAM-Iztapalapa
[161] Y.J. Yu et al., 2008, "Traceability for the Maintenance of Secure Software", Software Maintenance, ICSM 2008 IEEE International Conference, IEEE, pg: 297-306 International. Conference on Requirements Engineering (ICRE 1996), 1996
[162] J. Yoder, 2011, "Refactoring at the core of agile software development", ACM DL Library, pg: 51-52
[163] A. Zisman et al., "Towards a Traceability Approach for Product Families Requirements", Proceedings of 3rd ICSE Workshop on Software Product Lines: Economics, Architectures, and Implications, Orlando, USA, May 2002
[164] A. Zisman et al., "Tracing Software Requirements Artefacts", Proceedings of the 2003 International Conference on Software Engineering Research and Practice (SERP'03), Las Vegas, Nevada, USA, June 2003
[165] A. Zisman and A. Kozlenkov, "Consistency Management of UML Specifications", Proceedings of 4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03), Lbeck, Germany, October, 2003
[168] X. Zou et al., Phrasing in dynamic requirements trace retrieval, in Proceedings of 30th Annual International Computer Software and Applications Conference (COMPSAC06), 2006.
[169] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998.
[170] International Organization for Standardization, Software Process Improvement and Capability Determination, ISO 15504.
[171] NASA, "Preferred Reliability Practices: Independent Verification and Validation of Embedded Software", Practice No. PD-ED-1228, Marshal Space Flight Centre

[172] Center of Excellent for Software Traceability, http://www.coest.org/
[173] Teleologic, Teleologic DOORS, www.teleologic.com.products.doors
[174] SEI-CMM, htttp:www.sei.cmu.edu.cmmi
[175] SWEBOK http://www.swebok.org
[176] Xlinkit. http://www.systemwire.com/xlinkit.
[177] CORE, http:www.vtcorp.com
[178] RDT, http:www.igatech.comrdtindex.html
[179] Integrated Chipware, RTM, www.chipware.com
[180] http:www.agilemanifesto.org.iso.en

Chapter 11

Appendix

11.1 The interview package

11.1.1 The interview guide

Interview Guide

Master thesis: Traceability in Agile software projects

Author: Hoang Vuong
hdvuong2001@gmail.com

First of all, thank you very much for your participation in this interview.

I can make sure that all information related to you and your company must be ***anonymous***.

Your information will be recorded and used in my thesis, which will be sent to you as the final product before it will be published.

The interviewee is provided with 03 HANDOUT to answer relevant questions. Please pre-review these questions. Particularly HANDOUT-03

A/ Opening questions

1. How many employees are working for your companies?
2. How many software developers?
3. Which domain are you focusing on?

4. What is the typical size of one of your projects? Example:
   - number of developers participating;
   - number of lines of code,
   - number of requirements, time, budget, etc?

B/ Main Questions

I/ Agile practice

1. Can you say the degree you are using Agile and how effective you think of Agile? To answer this question by filling in HANDOUT-01

II/ Core purposes of Agile and Traceability

2. Do you agree or disagree with main principles of Agile? To answer this question by filling in the section 1 of HANDOUT-02

3. Can you express your opinion? about some software engineering activities by filling in the section 2 of HANDOUT-02

III/ Traceability types and directions

Each tracing direction is corresponding to one or more than one Traceable types. Can you tell:

4. What are important tracing directions corresponding to relation types? To answer this question by filling in sections 1-5 of HANDOUT-03

- Probe: Are there any other important relation types and/or tracing directions you want to add? Why?

5. Why do you think these Traceability types and/or directions are important?

- Probe: Why un-marked types and/or directions in HANDOUT-03 are NOT important?

6. Do you set any condition/factor to add Traceability to Agile projects?

    - Probe 1: what are conditions/factors?

    - Probe2: Why these factors are important?

    Factors can be: the type of project (i.e, security or medical sw systems); the size and length of projects; etc

IV/ Traceability mechanisms

Traceability mechanisms are a collection word including: tools/methods/frameworks/techniques, etc. Mechanisms help to achieve Traceability.

7. What mechanisms are you using?

- Probe 1: why do you think these mechanisms are beneficial?

- Probe 2: Can you briefly describe how to use your mechanisms?

8. Do you know other mechanisms that you do NOT use?

- Probe: why these mechanisms do not work in your organization?

9. How do your mechanisms WORK WELL with your Agile practices?

- Probe: what Agile practice do you think is the BEST helping to achieve Traceability? Why?

10. Do you think that mechanisms supporting for *traditional* software methods can be used for Agile?

- Probe: why and why not?

11. If you are looking for an effective Traceability mechanism, what outstanding characteristic should this mechanism have?

V/ Benefits vs. Challenges

12. Apart from purposes of conducting Traceability mentioned before, what other benefits do you think that Traceability could bring to your projects?

13. Can you say about difficulties when implementing Traceability in a Agile project?

14. What are main causes of these challenges?

- Proble: where do causes come from?

  (external source: e.g. from customer or lacking of tools; or internal source e.g. the lack of skills & knowledge or communication within the team)

15. Some team members considers activities such as: documenting artifacts, updating Traceability links, are "waste" or increase extra/administrative work in Agile projects without any economic benefit. Why these activities are considered as "waste"?

- Probe: How do you cope with these issues in Agile projects?

16. Who do Traceability activities in your projects?

- Probe: are people interested or resistant to implement these activities? Why?

C/ Closing Questions

1. Do you know if your company had workshops or seminars about Traceability in Agile for employees before?

2. Is it helpful that your company should have more discussion about Traceability in Agile in the future?

3. If you think that we miss something or not cover some aspects, please tell me.

That is the end -> Thank you very much for helping me and giving up your time

11.1.2 Three HANDOUT

HANDOUT-01

Instructions to give your opinions about Agile Practice

Basing on two perspectives below, mark your opinions in all practices of Agile listed below:

Level of use	*How effective you think the practice is*
1 = Not know about it. Thus, don't do it	1 = not effective at all
2 = Know about it, but only do it sometimes	2 = Relevant, but not helpful
3 = It is a part of normal process, thus do it most of the time	3 = Effective
4 = Central and always do it	4 = Very effective

	Do you use it?				How effective is the practice?			
	1	2	3	4	1	2	3	4
Pair Programming	☐	☐	☐	☐	☐	☐	☐	☐
Test-Driven Development	☐	☐	☐	☐	☐	☐	☐	☐
On-site customer	☐	☐	☐	☐	☐	☐	☐	☐
Coding standards	☐	☐	☐	☐	☐	☐	☐	☐
System metaphor	☐	☐	☐	☐	☐	☐	☐	☐
Small release	☐	☐	☐	☐	☐	☐	☐	☐
Collective code ownership	☐	☐	☐	☐	☐	☐	☐	☐
Sustainable pace	☐	☐	☐	☐	☐	☐	☐	☐
Planning game	☐	☐	☐	☐	☐	☐	☐	☐
Refactoring	☐	☐	☐	☐	☐	☐	☐	☐
Continuous integration	☐	☐	☐	☐	☐	☐	☐	☐
40 hour week	☐	☐	☐	☐	☐	☐	☐	☐
User story	☐	☐	☐	☐	☐	☐	☐	☐
Acceptance test	☐	☐	☐	☐	☐	☐	☐	☐
Common room	☐	☐	☐	☐	☐	☐	☐	☐
Product Backlog	☐	☐	☐	☐	☐	☐	☐	☐
Sprint planning	☐	☐	☐	☐	☐	☐	☐	☐
Stand-up meeting	☐	☐	☐	☐	☐	☐	☐	☐
Daily build	☐	☐	☐	☐	☐	☐	☐	☐
Sprint review	☐	☐	☐	☐	☐	☐	☐	☐

	Do you use it?				How effective is the practice?			
	1	2	3	4	1	2	3	4
Scrum master firewall	☐	☐	☐	☐	☐	☐	☐	☐
Lock priority within Sprint	☐	☐	☐	☐	☐	☐	☐	☐
Decision in one hour	☐	☐	☐	☐	☐	☐	☐	☐
High level design	☐	☐	☐	☐	☐	☐	☐	☐
Self-organizing team	☐	☐	☐	☐	☐	☐	☐	☐
Team of seven	☐	☐	☐	☐	☐	☐	☐	☐
Eliminate waste	☐	☐	☐	☐	☐	☐	☐	☐
Built quality in	☐	☐	☐	☐	☐	☐	☐	☐
Empower the team	☐	☐	☐	☐	☐	☐	☐	☐
Create knowledge	☐	☐	☐	☐	☐	☐	☐	☐
Deliver as fast as possible	☐	☐	☐	☐	☐	☐	☐	☐
Optimize the whole	☐	☐	☐	☐	☐	☐	☐	☐
Decide as late as possible	☐	☐	☐	☐	☐	☐	☐	☐
Amplify learning	☐	☐	☐	☐	☐	☐	☐	☐
Build integrity	☐	☐	☐	☐	☐	☐	☐	☐
Domain object model	☐	☐	☐	☐	☐	☐	☐	☐
Development by feature	☐	☐	☐	☐	☐	☐	☐	☐
Individual class ownership	☐	☐	☐	☐	☐	☐	☐	☐
Feature team	☐	☐	☐	☐	☐	☐	☐	☐
Inspection	☐	☐	☐	☐	☐	☐	☐	☐
Regular builds	☐	☐	☐	☐	☐	☐	☐	☐
Reporting results	☐	☐	☐	☐	☐	☐	☐	☐

HANDOUT-02

1) According to literatures, Agile has following core philosophies. Filling in to express your opinion (*1 = Strongly disagree; 2 = Disagree; 3 = Neither agree nor disagree (No idea); 4 = Agree; 5 = Strongly agree)*

	1	2	3	4	5
a) Respond to change quickly	☐	☐	☐	☐	☐
b) Reducing waste	☐	☐	☐	☐	☐
c) Deliver working sw frequently	☐	☐	☐	☐	☐
d) Promote communication in team	☐	☐	☐	☐	☐
e) Simplicity	☐	☐	☐	☐	☐
f) Customer collaboration	☐	☐	☐	☐	☐
g) Customer satisfaction	☐	☐	☐	☐	☐
h) Motivate team members	☐	☐	☐	☐	☐

2) Describe and express your opinion about some activities by filling in in these choices below:
1 = Don't know it at all. Thus, never do it
2 = Heard about it and did similar tasks a couple of times
3 = Know about it, do it somestimes
4 = It is an integral part of the development process. Thus, do it quite often
5 = It is extremely important. Thus, always do it

	1	2	3	4	5
a) Change impact analysis	☐	☐	☐	☐	☐
b) Requirement validation	☐	☐	☐	☐	☐
c) Re-use software artifacts	☐	☐	☐	☐	☐
d) Coverage analysis	☐	☐	☐	☐	☐
e) Tracking links among artifacts	☐	☐	☐	☐	☐
f) Software maintenance	☐	☐	☐	☐	☐
g) System debugging & verification	☐	☐	☐	☐	☐
h) Process compliance	☐	☐	☐	☐	☐
i) Organization learning	☐	☐	☐	☐	☐
j) Change management	☐	☐	☐	☐	☐

HANDOUT-03

Instructions to identify important Traceability types and directions in sections1-5

Identify a relation type corresponding to a tracing direction. Please carefully see notes or explanation below:

- Req = requirement
- Other = test cases, goal document, reasons for docs, etc
- STHOL = stakeholder, who makes the software artifact or element
- Element of a system = (stakeholders/requirement/design component/code/test data/ etc)

- Suppose that we have element e1 and element e2 of the system and 08 types of relations explained below:

Dependency:	e1 depends on e2 if the existence of e1 relies on the existence of e2. Example: dependency between 02 requirements, or dependency between one requirement and one design component.
Refinement:	this type is used to identify how a complex element can be *broken*; or how elements of a system can be *combined to form other elements*; or how an element can be *refined by another element*.
Evolution:	e1 evolves to e2 if e1 has been *replaced* by e2 during the development lifecycle of the system. Example: Req1 in the requirement document V.01 has been replaced by Req1.1 in the requirement document V.02
Satisfaction:	e1 satisfies e2 if e1 *meets the expectation or needs* of e2. Example: e2 (a design component) is used to make sure that e1 (a requirement) is satisfied by the system.
Overlap:	e1 overlaps with e2 if both e1 and e2 *refer to a common feature* of a system. Example: e1 (a requirement statement) and e2 (use-case) both refer to a common feature of the system.
Conflict:	this type shows the conflict between e1 and e2. Example: e1 and e2 are two requirements and they are in conflict with each other.
Rationale:	this type is used to represent and maintain *the rationale behind the creation and evolutions of an element*
Contribute:	this type is use to point out *who is the author* to make an artifact.

How to mark a box in sections 1-5

- You have 02 options for each box (marked box = *important*; un-marked box = *NOT important/un-relevant).*
- For sections 1-4: each tracing direction in these sections can be marked with one or more than one boxes, corresponding to one relation type. Example: Req->Req can be *dependency* or *evolution* or *overlap* relation. Thus, you answer is:

	Dependency	Refinement	Evolution	Satisfaction	Overlap	Conflict	Rationale
Req-> Req	☒	☐	☒	☐	☒	☐	☐

1) Tracing from requirement to other artifacts

Relation Types

	Dependency	Refinement	Evolution	Satisfaction	Overlap	Conflict	Rationale
Req->Req	☐	☐	☐	☐	☐	☐	☐
Req-> Design	☐	☐	☐	☐	☐	☐	☐
Req-> Code	☐	☐	☐	☐	☐	☐	☐
Req-> Other	☐	☐	☐	☐	☐	☐	☐

2) Tracing from Design to other artifacts

Relation Types

	Dependency	Refinement	Evolution	Satisfaction	Overlap	Conflict	Rationale
Design->Req	☐	☐	☐	☐	☐	☐	☐
Design-> Design	☐	☐	☐	☐	☐	☐	☐
Design-> Code	☐	☐	☐	☐	☐	☐	☐
Design-> Other	☐	☐	☐	☐	☐	☐	☐

3) Tracing from code to other artifacts

Relation Types

	Dependency	Refinement	Evolution	Satisfaction	Overlap	Conflict	Rationale
Code->Req	☐	☐	☐	☐	☐	☐	☐
Code-> Design	☐	☐	☐	☐	☐	☐	☐
Code-> Code	☐	☐	☐	☐	☐	☐	☐
Code-> Other	☐	☐	☐	☐	☐	☐	☐

4) Tracing from Other to Other (Other = test case, doc goal, rationales of creating docs, etc)

Relation Types

	Dependency	Refinement	Evolution	Satisfaction	Overlap	Conflict	Rationale
Other-> Other	☐	☐	☐	☐	☐	☐	☐

5) Tracing from Stakeholders to artifacts

	Contribute
STHOL-> Req	☐
STHOL-> Design	☐
STHOL-> Code	☐
STHOL-> Other	☐

11.2 Interviewed data by companies

11.2.1 Company A

Table 11.1: The transcript of company A

Question	Answer
Could you give general information about yourself, your company? No of employees, developers? Average time, or No of requirements of a project?	The program I am working for is 20people, no developers since we are deploying and configuring the tools. The tools all are for Ericsson use throughout the world. The tools for requirement management, tools for managing test case, test plan, tools for product backlog. In general, there are a lot tools involved. Currently, 7200 developers are using specific tools. We have 02 main releases per year, 10 Sprint, we are using Scrum. So that about 01 year a project. I think about 1500 requirements. We have 5 different offices, 03 in Sweden, 01 in Germany, and 01 Canada. We have common meeting in afternoon at 2pm of Sweden.
What factors to decide the level of traceability?	For each project, you need to take active decision. Thus no default layouts you can choose. Yes, it depends on project by project.
What are important types of traceability?	Full traceability chain from high level requirements to detailed requirements (for relationship type Refinement) to test plans/test cases and also to implementation stories in the product backlog/sprint backlog is of main importance. The feedback to management and product owners for which requirements that has been implemented, and in what version and when deployment to end customer will occur is really important. Here we currently use document type reports. The reports are named SOC (statement of compliance) clarifying which requirements that have been implemented (relationship type Satisfaction). SOV (statement of verification) on that the implemented requirements also meet the quality standards. During the work we also need to track any changed requirements even without a formal change handling, due to trying to be more agile. Hence we are also are setting up suspect links handling (for relation type Dependency, Evolution) in the applicable ALM tools. Meaning that if a requirement is changed, a defined profile set to what type of linked artifacts (requirements) that need to be verified if they are affected. The tools always trace who has written a requirement, story, test case or made which change etc (relationship type Contribute). So: - High level requirements =>detailed requirements (M:M) - Detailed requirements =>test plans/test cases (1:M) - Detailed requirements =>implementation stories in the product backlog/sprint backlog (1:M) The relationship types Rationale is to my understanding meaning the reason behind something, in this case a product for example. This is covered by the Vision statement in development projects based on Scrum, or a Roadmap for a knowledge area etc.
What types of traceability are not important?	Relationship type Overlap and Conflict is party handled today where the requirements are made up of models or by using GAP analysis or a graphical representation of all requirements for a specific product or release. Those are a bit more difficult to have a fully covered common process to handle for the whole Ericsson product portfolio at the moment. It's instead handled differently product by product or even feature by feature.

What tools are you using to achieve traceability?	We are using the same tools, based on IBM Rational JUST Platforms. Tools support needs for releases at quite advance Traceability. Tools support from requirements to test case, defect item, backlog item. Server based tools, web based client for most tools, you have common log in for all tools, you get 03 main tools. RRC (requirement management tool); RTC (tools for product Backlog management, CM part to handle code); RQM (test management tool: all test plans, test case, automated, so on). Outside main tool box is higher level main requirements tool called focused point from IBM rational. Form main requirement, divided between diff products, you shoot down requirements to specific RC project areas. This requirement project area either based on project or product or feature, so forth, etc. From this requirement project area, you can have several teams working. So they could either be set up as several teams as one RTC Backlog area ... or connected to same requirement area. That is the same for test management tool. You could either have o1 main requirement trace to 01 requirement and trace to backlog item, test case. Or you have collection of requirements connect to test plan, you can get statement of compliance saying for requirements (Statement of Compliance and Statement of Verification).
Your tools work well in Agile projects?	These tools are beneficial to our organization. Different options, but this is set up for us. But our unit just uses IBM rational and this tool is beneficial as we are not co-located in one place.
Do you know other tools, but you are not using?	Open sources, JIRA, MARS, team foundation server are, ReqPro NOT used for us. Because we aim at reducing numbers of tools to reduce costs. But we are freely to select tools so long as these are effective. Additionally, test management tool made by Swedish HanSoft, this tool needs manual integration.
What is the best practice of Agile to support traceability?	Daily stand-up meeting is BEST to support Trace in Agile as that is one time we meet, we can share information. Product Backlog work well to support Trace also.
Do you think that mechanisms designed long time ago to support traditional software could be used in Agile?	Yes, I think so as for tools described, they have diff processes set up if you are using lean, Scrum or more Traditional methods, you apply process templates basing what methodology you use. At large project, or company, it is unavoidable for mixing processes. Actually, we have 03 hierarchical levels for product owners; we need Traceability that is beyond Agile working processes. Especially for requirements since one school of handling requirements in Scrum way would say that we not need any, to handling requirements we have everything in product backlog ... but that is impossible for big companies like Ericsson.
What do you expect for the future traceability mechanism?	Tools should be quite sufficient actually, but features supporting collaboration need be further developed because we are not co-locations, we need really good help to able to work as if we work in all one place. Ex: in product backlog tool RTC, we have all stories; each story there is common thing section. There are a lot discussions going on for a particular story, should I interpret like this. Now I need help from you with this. So there is a lot collaboration going on. We have many Video meetings, few features to help us to gain screen shots of that story very quickly, e.g.2-3 seconds. Thus, tools should be further developed to support for collaboration.
What benefits of doing traceability?	Traceability not only fulfills requirements, but also important for marketing department when they ask to write marketing materials in new products because if you develop with Agile, you are not setting everything from start. Marketing department need to know what going to be changed, what need to be added? That is an example. I would say that general opinion that Traceability reduces needs of docs, so it benefits. Most people in program where I am working see advantages of Traceability instead of writing lot of documents. If you see Trace as the leaving connection between requirements and use story, so on, you have tools supporting this you are working, you can easily get overview using Trace links instead of reading and writing documents.

Can you say some challenges of doing traceability?	If you have too many dependencies, it tends to be large that is hard to get overview of. That could be a problem actually. But main route cause is often you slice deliverable in wrong ways.... you will have to go back and read up for them.... to get simple Traceability, dependency between them.
Who should be mainly responsible for doing traceability?	We have service responsible within a team. They are responsible for this area to make sure Traceability is added. We are trying not only cross-function in a team, but cross function individual as well. So that we do not have anyone assigned in the team that you are doing all the test things, this is not the case. That mean everyone needs to work with Traceability activities and they are interested. Of course, you need time to learn new tools. So the tool might be the problem, but not Traceability issues.
CLOSING: if you heard about formal seminars about Traceability in your company? Is it helpful to have one in the future ?	I do not think I see a formal seminar focusing on traceability at my company before. I do not think there is a need of seminar focusing only on traceability. But I think this is a part of introduction about Agile methods to new members. I have worked about 2 years with Agile and traceable issues. This is the big area.

11.2.2 Company B

Table 11.2: The transcript of company B

Question	**Answer**
Could you tell information about your company? e.g., No of employees, main products, No of members in a project	I have been working with 4 diff software companies. Currently, I work for company with over 40000employees; we do mostly in semi-conductor, software just for supporting only. 500-800 software developers working for 04 diff areas. The first is IT for infrastructure, write scripts or support for IT infrastructure product companies. The second is for marketing IT; they are responsible for webs, security, etc. Third is building tools since we are selling many semi-conductor chips. The fourth group builds tools, using our chips for design, we recommend chips and design. Our products for every fields as we are selling chips, semi-conductors: Telecoms, Phone, everywhere,, etc Currently I work for building tool group. We have different projects, a year may be.
What are important types of traceability?	I worked for 04 companies. Some companies are strict for process, requirement, document, specification. My current company, we not release software firm, we not have huge requirement for OK go ahead start everything from requirement to go all the ways down. We cut short all those steps, we move very fast. We come up ideas, try to come with the ways. Importantly, if you spend much time with requirement, document, all kinds of test verification, it will take a lot time. If you go ahead with minimum requirement you can throw out good your idea to publish, they try it. If they like it, you get feedback immediately. So it depends on situations. Some projects require very good docs, e.g. long-time project, we really care for docs: requirement, design, verification, etc that mean you have to plan upfront for all tasks. For short projects for marketing purposes, we ignore documents many, we just go ahead and build them quickly and throw them out. It depends the team nature, if the boss is really software guide, he can gain a lot from tools, process. If the boss is not I used IBM models actually. We talked about requirement, we keep track and docs (word, excel) you put all requirement, name them in certain ways. We do not use tools, if he wants a change, he knows exactly how to do. The boss said, it is expensive, we not benefit from doing tools.

What mechanisms are you using to achieve traceability?	Not really cause one requirement of our system is flexible for change. We just do manually, not need tools. It really depends on levels of your team members. We have much experienced members; we know exactly what to do. Something is obvious. Generally speaking. My team members are very experienced (20-30 years), when ideas come, they know very quickly what need to be done as they have much experience. They know how to turn things quickly. They need not to spell out; they cut short a lot, to get things out. We have new members, who are struggling what to do. So experienced members come to them, help them. So it depends on situations that we make decisions. I used some IBM rational, Clear Quest. I did evaluate some tools last years for change management or requirements. My previous company, we care a lot about framework, process. But not for current company as members know what exactly what to do. I worked with diff groups, and then I need requirement very clear telling me what exactly what need to be done, what need be delivered as they not understand the team. When there is a change, I need to see from both sides. But it is different with our team. I worked for 04 companies; we never go seriously to the process They not care for docs, who read docs (big laugh!)? It only is applicable case by case. We are not selling software to sell, but just support ourselves. We are diff with software house, who may care about process, requirement management tools.
Which agile practice is the best to support traceability?	We apply some Agile practice, people meet a lot when starting projects. Again, it depends on project and individual experience.
What are benefits of implementing traceability?	Sometimes people coming up with ideas, they quickly build and throw out to get comments from publish. Depending on feedback, now I have to change. They keep change of requirement; they never have fixed requirements, just minimum. When you work with Agile, one thing you have to watch is all requirements will be changed because every time you read, you get feedback from customers. To keep track of change, it is very difficult. You just let the public to tell you where to go.
What are difficulties of doing traceability?	It creates overhead, heavy routines to make documents and implement Traceability activities. The knowledge and experience gaps among team members. Solutions: we use communications like sit down and talk a couple of hours, team members get ideas and implement quickly.
Closing questions	We have 60 people in my office. In USA, we have 04 offices: in Taxes, Cororado, we have team in India. On global, we have office in Japan, Germany, UK, many offices in other countries.

11.2.3 Company C

Table 11.3: The transcript of company C

Question	Answer
Can you tell some information about your company? e.g. number of employees, focus of domain product, number of people in a project	We have 5000 employees globally. The whole corporation has more than 100.000 employees working on the globe. 80% products for other member companies within the group, 20% external: such as car industry, H & M company, some banks. 40 teams across the global for a project.

What are important types of traceability?	When tracing requirements to other artifacts: dependency and satisfaction. When tracing from design to other artifacts: refinement and satisfaction. When tracing from code to other artifacts: dependency and satisfaction. Contribute: that mean he traces who made this artifact or element. Apart from this, the relationship between ... capacity and work in Agile mode. When working in Agile you should know exactly levels of details of work you are taking during a Sprint because you can change.
What factors to decide the level of traceability in your project?	My answer is simple, the bigger and longer it is more important for traceability. Of course, more complex, I mean if simple and short project then it is easy for traceability, e.g. for instance: backlog you can trace them. When it comes to a very long project, there is some traceability for dealing with project time life. So right at the beginning, it is not possible to have everything as explored at available. Some detailed aspects can be explored during the project also.
What tools are you using to implement traceability?	We have tools, but we could not use tools fully cause of some bugs, and lacking of knowledge. So we have tools and use tools some specific degree, but you can manage dependency and kinds of relationship among diff objects: product backlog items, Sprint Backlog items, which can be traced. So If you have any softer, then main tasks automatically start. So that is level of details you can manage by using particular tools, but we could not use these tools because of some other commitments and time plan.
The tool works effectively?	We are using team foundation server, and this tool is beneficial.
How do you use this tool?	When using this tool, you have diff templates which you can actually use to clear product backlog items. When you come up details for implementation, product backlog items can be broken down into Sprint backlog items related to a specific Sprint. When it comes to Sprint backlog item, if you have a dependency, traceability to some other items to complete, you can define impediment and differ types of levels. When you define impediment or dependency at different types of levels, when you complete this impediment, you can come back to previous author of particular task.
What tools you know, but you are Not using? Why?	a lot of tools within Volvo group. But we used IBM rational before in other projects. Right now, I do not know and I only use team foundation server for current projects. For dot Net and MS area, team foundation server should be used. For web or java area, I not know may be IBM rational. I think this tool work well but rather complex actually, to be honestly, we not use this fully cause of lacking skills and we not have time enough. I think that we need to improve this situation. We may be using more tools, but I do not know which specific tool for a specific project.
What Agile practice is the best to support traceability? Why?	The first thing is obviously, the product backlog. Without it, it is impossible to have traceability. Collective code ownership is the first because self -organization /motivating, people understand the importance of things when you work. In Agile, it is very important you monitor your own tasks and compliance to those. So it is very important you know you have committed for particular task. Having Backlog at the beginning, it will reduce specific scopes or also give Scrum master specific tools or probably ... to control that scope of the project. Empower team as you need to interest the team it come to fulfill the tasks. Others: continuous integration, TDD, etc can work well for traceability.
Do you think traceability tools supporting for traditional software methods, can be still used for Agile methods? Why?	Tools are supporting members in Agile. But without tools, you can do traceability. It is more with experience than tools when handling traceability. Tools just simply the way you are doing. It not so dependent on tools. I think we are good example for our tool (team foundation server), that is quite good. We should use this more, but still very complex, we did not understand this tool fully then we skip some parts.

If you are looking a future tool, what outstanding feature of the tool should have?	Big laugh:) that is very easy to answer: at least for me, I prefer tools that easy to learn and just support you partly. So you can easily start, not need fulfill every need, 80, 60, or 50% rather than complex tool then you do not use this tool because it is so complex to learn to use.
Apart from benefits as mentioned in HANDOUT-02, section 2, what other benefit can you gain when implementing in an Agile software project?	I think you forget important thing: economy and delivery on time. That is really thing, even in Agile, you not discuss those terms because how far or how much you deliver so far in %, 5 or 60% of the backlog? Where are we? And how much money this spends? Those are always questions that Customers ask. It is very import like we are called by stakeholders; we need to know what is percentage that we have, it is very critical to trace. during the project execution. Secondly, when working in Agile, controlling change/requirement is very import. Otherwise, it is very difficult to complete any Sprint if you do not control changes. When it comes to change management: in between Sprints, you always have delivery at the end of Sprint. During Sprint, when you have quite a few change after user description or after more implementation. It is very important, you define plans to implement that during Sprint or after Sprint and control that implementation. We are more focusing on delivery in term of: what, when deliver and economy cause we are project managers. If you have the product owner, probably, she does not discuss these terms, and depending on whom are you talking to.
What are difficulties when adding/conducting traceability in Agile project?	One difficulty is when working in Agile, if you have big project like this when having quite few members in the team. The first principle is self-motivating/organizing team. It comes to implementing Sprint, the difficult is we have tasks located for members and expect team members complete and commit fully. It is difficult sometime because it is not get all pp at same levels. Some people strictly make their commitment, some may not cause of their limitation. When they work, they do some work commitment or people do under-commitment. It is difficult when you have big team as if 5-6 pp it is easy, when you have teams at 30-40 sites you have Scrum of Scrum, it is difficult to monitor. It is hard to control what they commit, when they deliver.
As the Project managers: how do both of you cope with difficulties?	First thing, in Agile the meeting at the end of Sprint, it actually helps to understand, self evaluate team members. Basically, having self-respecting team meeting, people like questions: what have you done? What improve next Sprint? It is important to have clear: what is the definition of done or completed for each task? So that members to know very clear at the beginning. We focused on deliverables that we have to trace. We break down requirements to specific deliverables that we are tracing, evaluating. In our case, we have business analysis and test teams to support and keep track of these issues.
Who do traceability activities in your projects?	project managers do traceability, try to find out we did what we suppose to. We have Scrum master, business analysts, tester team all participate. We used use-case to go through use cases, etc to answer have we implemented everything? We are semi-Agile, not fully Agile.
Closing interview	Yes, I heard that we had formal discussion before. We think that it is helpful to have formal discussion about traceability in future.

11.2.4 Company D

Table 11.4: The transcript of company D

Question	**Answer**

Could you say general information about your company and yourself?	I have been working 20 years. The last 10 years I have been the managing director, rather broad responsibility. Before that, I wrote code, PM. My company has 32 pp, 29 are developers, focusing on XP method. We focus on Telecoms, Automotive sectors. Typical size project: 10 people working, 6months to 02 years. It is about 20million SEK.
What are important types of traceability?	Traceability from code back to requirement or perhaps artifacts to requirements. It is essentially one of great thing about Agile methods is you get rid of lot software do not really contribute to. Being able to maintain code and constantly re-develop it and make them nimble and faster and management, you need know why certain features were there from beginning. That is you did not remove essential functionalities when you simplify system. We did not only remove unused functionalities, we removed functionalities that are better performed in other ways. Knowing why you do something is really important. It is good that Agile help to gets cores of what customers need, because you involve customers. Typically, in V model you will have someone spoke to customers, but not really users, gatekeepers. From that, you put up business requirements that become technical requirements then become code, design. And often, from beginning you not know what customers really need as you not speak to customer, managers. If you develop system for call center, you actually try on call center personally; you get the core of what they need. That is process develop itself actually set up requirement because you try early versions on users and you show managers to involve them. So it is very important that the requirement is started with good ideas, but lot of requirement is really becoming explicit during the development. To combine that with ideas to keep code clean, you need to know why you added codes in certain steps because someone at call center really want to have, etc. We handle this link by writing in code why it was there and who made this, e.g. Peter said that he want this, he wrote in code. So the link between stakeholders to artifacts is important or "contribute". We wrote that in code, not requirement document. We wrote all stakeholders involving like: who made this, who reviewed, who commented, etc because we want to know that if we change later, the one who said that the last version was fine. If you do not know who involved in, you cannot trace back to diff stakeholders. When you have 01 stakeholder, it is easy. But if you have some stakeholders, like different departments: marketing department wants to trace to costs, management department wants to see where we spent out support time, etc. It will be difficult. So that when we change or simplify, we could easily trace back to say.
What traceability types are NOT important?	I would say the small thing that you discover in the automated testing. You do not need to spend a lot of writing. The testing system will alert you if you make mistake. We did not focus on small stuffs like that, but rather in general. We use tools to help us to do quickly.
What factors to decide the level of traceability?	It is essentially if it is just one stakeholder, we put less effort on Traceability. If it related more stakeholders, we put more effort on trace. For exam: if you would have one line of business using software, you typically have the manager and user (02 different stakeholders). That will be management and both of two stakeholders involve. If you have several lines of business involving, then it will become really important to have firm grips on Traceability. Otherwise, you will run around during one stakeholder want without considering other stakeholders. As Agile process, you listen the user, you do something, you listen manager, and you do something. If you not know why I did this from beginning, when speaking to next stakeholder, you just run around, it will never end. For project to complete within budget and time. You need trace requirements to relevant stakeholders. Otherwise, development process never end, and not able to satisfy the code. You will end up with unmanagement software product.

What traceability tools are you using?	We used tools for Traceability. I am not sure I remember, but tools are not from big companies, they come from small companies. We use a lot of open source, we also bought traceable software. We looked at IBM rational. The thing is that since a few years back no single tool could handle all. At that time, we pick some part of this tool + part of other tool + part of other tool, etc We had to combine tools to support our needs. We had database file maker for tracking and managing requirements. So we used tools for Traceability.
Do you know other tools that you are not using? And why?	I know Rational, Clear Quest but we do not use because they are not good enough for us. May be we may use them as they are getting improvement. At my time, these tools were good for really big project, but it was cumbersome for our size, it cost 10 millions SEK. So it was impossible for us. The point is the cost/benefits. It is essentially that we had the database called file maker, we different functionalities added on top of it.
Your tools work well in Agile projects?	We made these tools to work. We put efforts making them work for us. We bought part of functionalities, we used open source. Then we combined for what we want. Thus, I thought these tools work well for us in Agile.
Do you think tools designed for traditional software could be used in Agile?	Well, we did not use them because we thought they did not fit the bill. I mean they did not provide enough for us to use. That is why we used open source and little piece to combine. Yes, you can use tools for traceability. At the time, we do not use standard software for Traceability. I think traditional tools can provide some values, but process are so different, I have not looked these tools for years. It was not good enough for uses. It is essentially that you can use any tool for anything if it will not take workload that come from using tools. But we saw clearly that it was not good to use traditional tools for Agile.
What outstanding feature of a mechanism are you looking for in the future?	Essentially that the ability to link between all levels, from business requirement, technical requirements, user input, project input down to design, code, etc and they are tracked. I mean the connection to all because in daily work, when you change something, developers want as the tools help to identify to whom should I speak if I want to change this? to which document do I need to turn to check? why this is done and where it is? Essentially when all come down to whom I should speak? what should I check before I change -¿ that is benefits of Traceability? That I want for future tools, which help trace back to business requirements and why this process is need? who said this need? so that you can check it up. The tool should look like the 'Tree'.
Can you tell some benefits of doing traceability?	It would be business development then because often project come back to haunt you (laugh :). But you do project, you deliver, they use it. If something is changed in environment, e.g. supplied change, change in safe process, organization change, etc when you want to re-use project, and it is truly benefit to able trace back to formal business stakeholders. Obviously, you check who is the stakeholder for this process. The larger the project the more stakeholder they are, the more risky when they want to change something without knowing who is stakeholder. Essentially, I would say for requirement traceability would be like service subscription that business manager should subscribe, like you buy the insurance. Additionally, Traceability helps to see the progress of the project.

How about the difficulty?	I would say then it is not as difficult as it appear to see at first glance. Before I did, I assume that it would be much easier to keep track of requirement in traditional software methods, but it turns out that Agile methods actually involve users and managers, and make easier to pinpoint to a certain requirement or functionality to certain stakeholder. I done V model project, it looked easy at surface, but since you have typical situations with business requirement and business clients you never speak, users you never speak to, you have gatekeeper ask your customer company. Everything point to gatekeeper, but gatekeeper are not source of requirements, they just challenge. Agile methods break barriers to the connection between buying company with deliver company, between people who actually wrote code, e.g. Peter and Lars who want this way. Actually, it makes requirements easier to be traced back to source. It is important part from my opinion. On low level, it is difficult to keep traceable things because everything is change always takes place. But on high level, it is really... really important to have Traceability, it is easier with Agile than with V model. So it is difficult on low level because you change a lot. Essentially in perfect V project, you write Traceability once, that is really easy on low level e.g. this is red, this is code, that is easy. But it is more difficult in Agile because you change all the time when you talk to stakeholder, and then you change software. But on higher level, it is so truly to keep Traceability, and it is easier with Agile because you got pass the gatekeeper. The Agile methods force the buying company to allow access between developers and project manager into actual organization, not just gatekeeper, but managers and users. The gatekeeper is ”waste”, they are bad for business.
What Agile practice is the best to support traceability and why?	On site customer involve users in actual development and TDD because where we pinpoint we wrote all test case before we code, in test case I would refer to whom said what.
How you cope with difficulties?	Developers develop tails for traceability as they are ones to re-write code if they did not have Traceability. You know that I told you that running around when you have a lot of stakeholders, you speak to first stakeholder, he tells something, you speak 2st stakeholder they said something, etc. So you keep just running around, you not keep traceability you never finish. Thus, if you did not do Traceability, you will have to work many more hours.
Who should do traceability in projects?	I think all members should participate in Traceability activities. But mainly are developers because they really need to do.

11.2.5 Company E

Table 11.5: The transcript of company E

Question	Answer
Could you introduce about yourself and companies: e.g, No of employees, developers, how long for a project?	Totally, the company has 1700 people. My current department has 70 people, the group has 60 companies. I cannot know how many developers for the whole group. But my company is 35 developers. I recently switched to other areas. Before, I primarily worked with Telecoms. Now I work with all areas and working with different companies as Consultant (Telecoms, Aviation, etc). For one project, 10-15 people, and 6 months/project. A couple of hundred of thousands up to ten million SEK per project. I work 15 years for my company and 05 years working as project manager.

Could you tell what is important type of traceability?	Stakeholders to artifacts and Tracing from requirement to other artifacts are most important to be able to have a proper commercial motivation of work done. I would say you need to trace everything, from business activities to client. You need to show what has been done and why?. Now I have a co-worker having big Agile project where customers change requirements over, over and over and do not know why budget is not the same as at the beginning. They totally re-did everything. I want to know for each user story what is the exact cost? If you do not keep track of traceability, you cannot motivate all things that have happened with projects to talk to your clients. You need to keep track to trace who add for what and why? how much resource for this change ? what decision has been made? why? keep track of things like what customer actually said to me? If you have not have papers tracing on it, you will get trouble with.
What factors to decide the level of traceability?	Not formally for conditions to add traceability. I think you should do to make assessment on basic levels how different the clients could be?, which kind of relation you have with customers?, how clear with two sides?. If it is simple project, e.g. you move a system from this environment to that environment.... but it should be one to one. Perhaps not much un-clarity. I want new.... sale channel to support my customers, where you have marketing department to change marketing every minute. It is different how to handle Traceability aspects of between projects, e.g. size, type, etc. So it is more case by case, we did not have formal process to decide for a project, very flexible and depends on projects. From personal experience, so members use different ways to handle Trace issues (papers, phone, email, tool, etc).
What mechanisms are you using to achieve traceability?	We use some models of Agile, e.g. product Backlog, MS. excel sheet, email. We also use a specific tool JIRA. Some of my members use Team server foundation internally. When it comes to handling Backlog, it is more formal thing about specific requirements, we use those kinds of tools to customers to make priority and keep tracks.
Your tools work well in Agile?	Depending on projects to decide using tools or manuals. We also said some tools are not effective with some clients. If I look back to my background, my managers told that you should use this tool for handling requirements, this tool for reports, this tool for communicating, for change request, etc. You need to keep track of requirements somewhere. I do not have ideas if it is more effective to buy tools or use open source, or develop. It depends on a specific project.
Do you know other tools, but you are not using?	I think: MS team foundation server, IBM rational, Clear Quest, Share Point. But some teams at my organization may use as we are big Consultant. But my big clients, e.g. Ericsson, are using these tools. I cannot use all tools because I need to choose among tools, so we do not need duplicate to save resources.
What is the best practice to support traceability?	I think it is TDD as a good quality because I see it from my experience from selling fixed price projects, which mean our client tell us that these are requirements and ask how much to develop this system (fixed price), then we go through requirements, we deliver it, we take volunteer functional search, etc rather just saying OK I am selling you per hour, I work for you until done. It is important to have efficient way when you have not a lot of bugs. Thus, I believe TDD as well as continuous integration as we are doing as the best ones. Testing at all levels will solve problems. Some developers naively think that they have less levels of needed testing techniques to do things. That important thing to force people to do. I think small release work well also. It is not always possible and changeable in relation to, in some ways it is risky to have overhead, you have to release work more but big bang is over bad. An example: project 8-9 months, probably more efficient on our way to do in one release, but it raises risks too much, so we did two drops: one for setting infrastructure and the other for adding more business functionalities.

Do you think tools designed traditional software can be used in Agile?	You have tools that have base functions. You make some adaptations to new processes, like you have been following methodology. We have this and then you model, configure to support that process. For instance: at big Ericsson or Volvo, you could re-use these traditional tools to re-model, configure. Or you could kill and set up new tools. I do not think the matter is tool, but it depends more on how you implement works in your organization. So I think old tools are good business for ones to re-develop, to update to support new concepts, and so on. I am afraid, however, it is easy to answer and depends on situations.
What outstanding feature of a tool you are expecting in the future?	I think you need to be able to set up tools to suit what you want to do with it. It should be intuitive to developers, if you bring more new members, they need to understand what to do, what to get. I think it is more how you work with tool, how you promote internally all things rather than a specific tool, it is my gut feeling. You can make anything work as long as you adapt what you want to do. Some people are happier and some may be not.
What are benefits of implementing traceability in Agile projects?	Like I said, to me it is always to be able to have clients, you need to show what have done, why done, who added for this, why decision has been made, what impact it had not you escalated , etc. It come downs to commercial aspects of interacting with client basically. I am looking those from person, who is responsible for handling clients, who have business relations with clients. They want stuffs all the time for sure, but when it comes to pay, they do not want to pay for all stuffs. You need to have your motivation clearly, you keep stuffs together.
Can you say some difficulties of doing traceability?	Adding overhead, taking time from producing. Sometimes, Trace is requested as deliverables, sometimes you need to document but main deliverable is actual software you are making. So it adds costs and overhead for team members. You need dedicated resource for members to handling communication and requirements, all things. Some people say traceability is waste because if you are developer, you focus on finishing jobs, you do not see that someone need commercial discussion with clients, you not see if I do this today or I just want my module like this, e.g GUI, now I need to fill out messy documents as someone asks me to do. So that is waste for me. Then a manager talks to you: I need be able to : control, motivate, explain, trace, see everything, etc.
How do you cope with difficulties?	I think you need have good company practices, you need educate staffs to follow them, you need a good tool so motivate them to use . Ex: from my experience, we had internal course about methodology we were using, then we had PM course, that gave early skills to some of prosperous developers, and trying to infuse them with company culture where they understand importance of things, like what will be consequence if you have bug in the end of the process or in the production ? you scuttled up the bill cycle the for the client, and he lost 30 million SEK in revenue, that not good. If you caught these bugs in Unit test, we lost 20 minutes of fixing bugs in your code. If you caught bugs in system test, we lost 3 days in writing real releases. If you caught bugs in integration test, we lost 3 weeks for coordinating for changes, etc. If you need to infuse right values to your staffs.
Who should do traceability?	I think you need someone to lead the project, scrum team, your organization form. You set the levels, it depends situation, not have fixed process. Someone needs to set levels for situations. I think it is up to the one in project management positions, e.g. Scrum master, etc. she reports everything. Or you need to see what do we need and Scrum team need keep track how we are interacting, and we have project improvement manager, who needs to say that I need to have information of this and this is what is needed. Then for sure, you need to have basic methodologies that match somehow.

Closing questions	Probably I heard that we had discussion about traceability in Agile. We have 60 companies. Some of them have been really working with a lot of Agile things. People go out to teach people how to work with Agile - organization teaching. For sure, we had discussion before. It may not be top priority to have the focus on seminar of Traceability as it can be added in other seminars.

11.2.6 Company F

Table 11.6: The transcript of company F

Question	**Answer**
Opening	We have 20 people working and 16 software developers. We focus on E-commerce software products. For one project, 4-6 developers and about 100-200 requirements, around 2-3 months project, it is worth around 1-2 millions SEK. We charge per hour, not fixed price.
What factors to decide the level of traceability?	No, it is same size for most projects. Since we work in fixed domain, it does not diff much for one customer to one customer. We built sys foundation, platform for building E-commerce system. When foundation is already built, we build customization for each customer. The main part of project is finished before hand, it is easy to know that the amount of time to develop a system for customer. I do not think we set conditions.
What is important type of traceability? What is NOT important?	In some test-driven development (TDD) frameworks, you pick a requirement and, if it's well written, just split them into fractions. Each is used as a method name in your code and describes prerequisites, actions and expected results. If all tests are written before the implementations (TDD-style) and the test cases/scenarios are directly based on requirements, you will have a very nice documentation of what the system is expected to do in these scenarios and you will never end up writing code you think might be useful later on in the project (this is good because you won't be confused by your own code when skimming through it a year later and thinking "why is this here? Is it important?"). Since you write all tests directly based on requirements, it is easy to tell from the test code what the system is supposed to do and if a requirement is changed you will notice what you need to change in the code to satisfy the new test scenario (which you rewrote based on the changes to the requirement). In reality, however, it doesn't really work that well, since a lot of code/projects were built on other (or no) foundation like TDD. You basically have a code and the change of a requirement is specified in not so specific words. An example: "Can you change so that it is possible to use more than one gift card for a purchase on our website?" This is first of all not directly aimed at one specific requirement, it touches a lot of other requirements too, and the only way to learn this is by reading thousands of lines of code or in best case asks a developer who's been on the project for quite some time to help you understand. However, making changes like this when you can't really trace how a change ripples through the system almost always end up with poor satisfaction when the customer is performing the final tests - in most cases you have missed some aspect that you didn't realize was tied to the requirement you are changing.

What mechanisms are you using to achieve traceability?	We have one tool to keep track of requirements to see when they are delivered to customers. You can set status for each requirement (finish, done, deliver, accept, reject, act). For other thing, no, it is only tool for requirements. The tool is named Pivotal Tracker. If something is change, maybe we have time to update. Most of things are specified in Excel. They do not have links between them. They said this feature need be done for something. If they need update for change, member knows how to fix by himself.
Could you describe how your tool works?	With this tool, you can update for each item/req. It is webpage tool, so customers can access the sys. Thus, they can see our progress when they check it out. When there is change, we discuss in team. Basically, first you create specification for system, you decide things, and you break down requirements to items. You add to system backlog, and then you write in Sprints. You start working with assignments for developers, which are assigned by managers. If any change in these, customers access the system to change. I will see the change and see what has been changed and and understand what I need to do.
Do you know other tools, but you are not using?	I do not know other tools for traceability. But I heard about Kanban board, you post notes, you write requirements on board, and you write on columns (status of artifacts). You move around, it is kind of same idea. I think systems like this there are some. But I think they not help Traceability really you miss these links. You cannot see how things are connected.
Do you think your tool work well?	I like this tool as it has nice Interface, it is easy to access for both developers and customers. You get notifications by email if any change in your task. So it is good way to manage your work. This tool does not depend on Agile practice or traditional method. It just is a way to organize things to keep noted about what should be done, like ”to do list”. I mean it is very close to V model what we are doing now since we do not have any requirements accepted yet by customers. We are already to deliver, so it is really hard to work in Agile way. However, there is something dynamic to do, it is not strict system. I mean we mixed between automated and manual. There is a problem as we developed 03 Sprints before any testing was done from customers. So basically, we have deliverables 80 requirement before anyone looking to see what has been done. So it Not is Agile anyway, it is possible to utilize better in a more Agile way. Though I have never done big project, I can imagine that it is very hard in a big project because more risks when something will change.
What practice as the best to support traceability?	Not matter if you are Agile or more Static way (traditional method), TDD work equally fine, but it matters a lot for architecture for the system. TDD is great for supporting Trace because it is a nice way; a lot testing frameworks are for kinds of TDD related. Ex: you have a requirement saying when you press A button, Field B should be empty and Text C should be updated. You can break it down easily to test scenarios by TDD and implement software from that. I think TDD is great if you have right framework and if you have chance to decide the architecture of sys from scratch. I mean it is rare situation because it costs a lot to develop from scratch. You always depend on LEGACY system. But TDD is very good think for Trace because you are very care to see what has been implemented for what requirement, so on. Common room is also good thing for Traceability activity. You get very quick answer for fixing change, updating links. Additionally, Product Backlog is good as you are working in Agile.

What do you expect for a future mechanism to implement traceability?	I think TDD used in right way. If you have problems with Unit test or test in any way, from my experience of web system, if possible to test all things up to user interaction via TDD, I think this will be most optimal way. You have each user story, you base test on user story, you these connected with some kind of numbers or something, to make them linked, it is hard to describe. It depends a lot on what kind of system you are building. If you have clear starting point, it is easy to trace to see what I have to implement to satisfy that requirement? You can see exact my answer code for satisfying certain requirement.
Can you tell some benefits of doing traceability?	I thinks biggest arguments is sw maintenance. That is one affecting company developing sw and customers. So software maintenance is the most important. It is big problem not having tracing links. It is big risk. I think it should be something for organization, who arrange and make time for this in the budget. E.g. you have to spend 02 hours documenting, I think developers will do. We write small things in one place, we try to keep track which person working on what project and what task that person has been working for the project, so you know who to ask. But if that person quits, it is impossible. I have not seen this case. I see relevant points to do detailed docs, update links. But we have no resources to do.
How about your difficulties when doing traceability?	The problem is customer sitting in Canada bank. Our customer in Sweden is a consultant and the bank and everything is 02 steps. I do not know how they keep track of change. For developers, it is not necessary actually. The challenge is time; it never makes room for it in budget. It may be impossible to explain with customer why you need it. Ex: we build web shop, customer have not so much knowledge about IT, when we say we need to keep docs, or maintain software and you will need overhead as compared to original plan, e.g. 20% time plan to make sure traceability activities. You need a lot extra time, so it is very expensive. Customers say why do not doing perfect from beginning? Why am I paying for these stuffs that I do not see any difference? When developers have to write documents for other readers, it will take a lot of time, much more than you write docs for yourselves.
Who should do traceability do you think?	I think that customers mainly do traceability as he is the one to pay for this Trace. Even developers feel happier to do traceability so long as it is specified in budget.
Closing	May be, it depends on technology, software is very important, everywhere. I believe people would understand the need. People see traceability is good idea, but I think people do not have time to do or doing it in proper ways.

11.2.7 Company G

Table 11.7: The transcript of company G

Question	Answer
Could you tell some information about your company? ex: No of employees, main product, No of people in a project	Around 100 employees work for my company, including 60-70 developers. Domain products varies, but Telecoms is main domain. I work in this company over 4,5 years. I got assignment to work with Sony Mobile, with 100 people for a project. The project lasts for average 1,5 year.

What are important types of traceability?	I think important to trace from requirements to some kinds of verification. When you have requirements, you able to do test cases, verification and verify that requirements in sections are fulfilled and I think that is the most value. Again, you need trace from problem report to verification as well. So everything you decided that should be done in project no matter of which kind artifacts. You able to trace down to verification. Within project, most likely to trace requirements problem report to deliverables, what requirements deliver? When? And where?
What are NOT important types of traceability?	Tracing down to code level, line of code, test case, etc. It is too expensive really as the effort to keep updated is too large. If you have good tools, it could be valuable to trace details. Exactly you need to do automated if you want detail traceability.
What are factors for you to decide the level of traceability?	Yes, I think the larger the project is, the more important Trace is as if you go to extreme project in one Agile team I think the most knowledge about everything in project is captured within the team in the way the team think that they give most values. If they think that it is valuated to document things, they will. But if Now team fell they are controlled, we know about when we have do what, the need of trace in that cases is not big. When you scale up and you add on more teams, then the need of Trace increase quite fast because when you form strong teams, self organize team, you put the teams on internal communication and depending with in the team ...but that is done on the expense of external communication from team to surroundings. So the communication between teams in Agile organization is (h) they communicate less than ... uh... if you have weaker team. Stronger team communicates more internally but less communication external. If there are security, it not really matter if you are Agile or not, there are number of factors that affect the need of Trace. I also see that when I teach CM course, different industry have a lot of difference in need have trace. E.g. you compare game software, phone, pc software etc and compared with air plane, medical industries - the need of trace is much higher on those, more long -live, security based systems like airplane and medical system, like that.
What tools are you using to achieve traceability?	We have some scripts that help developers; we do not have frameworks for doing traceability. We have trace with our build tools, adding methods, scheduler tool, to build system that creates trace. The rest is mostly self-developed tools, something like that and enhance the environment to create Trace. We are using Clear Quest tool for problem report handling. But Trace tools are developed by our self for gaining the t information from problem report tool. So it is more we reduce database, I mean information is stored in Clearly Quest but the access is self developed.
Do these tools work well in your company?	These tools are benefit a lot. We do not have norm, time to spend on tools, but it increasing quality week by week. We are doing continuous improvement for tools, so it is getting better and better for our tools.
Do you know some tools but you are not using?	A lot tools, laugh :), it depends on what kind of tools, Scrum Work, On line tools to handle user stories. GRAP, REMIND, Rational (Clear Quest, Report), New Rational Team Concert IBM, IBM bug, a lot. The main reason these tools NOT work for us is we need to develop our self for our own needs. I think that is part or totally that there is strong culture for developing solutions by our self rather than buying. If you buy tools, requirements are both installation and training cost, maintaining cost, support costs. I think that we are feeling you do not gain that much form buying tools. In many case, you can do really slim solution by yourself.

What are best Agile practice to support traceability?	Our tools work fairly well, yes. We have strong support for continuous integration, collective township code, small release, strong support coding standards. We do not have strong support for built in and sprint review. The more people oriented Agile practice, we do not actually have. I am not sure if these are bad things. I found that being face to face and having white board work effectively. You can avoid tools for those tasks. We try to keep the team on 01 site. I think the BEST continuous integration for Trace really because when you work with continuous integration, you get small batch, small change batches. That will give better Trace to what change made for what. You start think more about what method should we work with, store for it to able to indemnify that works.
Do you think traditional tools could be re-used in Agile?	Yes, I mean tools for trace in traditional can be used for Agile. I think most traditional tools work fairly well with Agile practices, of course there is exception. Because they mix agile and traditional development methods, thus tools can work for both.
What outstanding feature of future tool are you looking for?	It should be flexible and easy to adapt to process, ways of working, and the team and if possible, it adapt to each team at least some aspects. But Flexible is key thing for being successful.
Can you say some benefits of doing traceability?	I think a lot of what you do when you add traceability like buying insurance. That means you want, e.g. if you not hand out errors, you will end up with errors. It make easier to make route causes analysis, to fix problem. It also is good when doing customers supports, when customers confirm situation with problems or questions, it make a lot easier to ... questions. It will give good integrity picture. If you have good Trace, you have a lot easier to give that feeling to customers, they will get have good system, you know what you are talking, valuable supports when customers have questions or problem. They are hard values. Soft value is you give the better impression on company.
How about difficulties of doing traceability?	Yeah, challenges are often almost kind of manual way to deal with change. Then, always information is get better liability because people enter information with manually, it cannot sure they are correct, you lack of trusts. If you do not give see the full picture of values of traceability, and then it is difficult, especially in Agile when you focus on delivering values. As long as you can provide good arguments, good overall pictures, they are less problems with adding traceability.
How to cope with difficulties?	to educate team members, reasonable explanation, you have to be honest, work with transparency.
Who should do traceability?	I think that everybody needs to do traceability. If I was, I do not want specific person do traceability work, I want every have information need to provide and work with traceability activities.
Closing	The company has already seminar about traceability issue in my company. No, not need separate traceability seminars, that should be integrated with development process. Presently, I am teaching CM with different organizations, companies.

11.2.8 Company H

Table 11.8: The transcript of company H

Question	Answer

Opening: some information about you and your company?	I've been working here 18 years. My responsible: system design group manager, driver of requirement handling, setting methods for the department in Goteborg. We are about 1000people, I represent Goteborg site department. May be 300 developers. Domain areas: Radio frequency, gets detection, measure, estimate aircraft position, etc. Small projects 10 people, big project 200 people, small projects 100 requirements, large around 5000 requirements.
What are important types of traceability?	We focus on 'refinement' type from customers requirements ->system requirements ->subsystem requirements, down to system elements. That is Traceability. We have database requirements, then we have other docs describing design, which is not in database, these docs described how requirements should be implemented in specific design. Then we have "rationale", so we think important to describe why you change something, why you implement in a certain way. We have lot of requirements, it may be difficult to have 'dependency' in actual tools as it is very expensive to maintain because there have so many connections. We use descriptions in design docs describing dependency, we have communication between engineering keeping knowledge in head to share, discuss how system working. They have requirements database to see how different requirements are traced and much knowledge of design to see how things are implemented and then they have to knowledge to see dependency, it is so expensive to have all Trace in Matrix because you do not need to see these patterns, and how to connect them. Thus, you have formal connection in tools or you have knowledge and docs to actual traces. I think evolution is also important cause to transfer knowledge from one engineer to other as you can make decision on something. It is import to keep knowledge about change about why we implement such things. Satisfaction is import as you have to satisfy requirements and design at upper level. You make sure you break down sys when building it, you fulfill customer's requirements. Test document is import for tracing We trace from requirements to and verification.
What types are not important?	I am thinking about Tracing, but not important for conflict type as we do not trace, you have to review, inspection to find conflict. Also overlap type as we do by inspections to find overlap cause something you need to correct docs or whatever before doing tracing. We know they are important but we not trace in that way.
What factors to decide the level of traceability?	We have traceability between customers and system requirements, subsystem requirements for all projects. However, not all in DB tool yet I think it is better to implement sub projects. But we have requirements to do traceability. For flight safety reason. There might be trace from requirements to line of code. So we consider factors to add Traceability level.
What mechanisms are you using to do traceability?	We use dimension RM. We also use DOOR for handling requirements. We have tools, but we are on the way to be better, not all engineers are using these tools because tools are complicated to learn and use. Now we are trying to simplify the use and to get better GUI, versions of tools. These tools are beneficial to our company.
Can you tell some tools you are not using and why?	Requisite Pro, IBM Rational, Focus Point. We might be using Focus Point for some requirements when you select most important requirements for priority before they are actually put into project for comparison between requirements. I think we did comparison between tools and we made decisions basing on different factors to select which tools. I think that DOOR is been upgrading to 2014 for better. They are simple and they are suitable for costs. The whole SAAB group to make decisions which tool for consistent use within group.
Your tools work well in Agile projects?	Dimension RM and DOOR work well with Agile practices. We are sometimes using pair-programming, TDD, on-site customers, collective code ownership, refactoring, daily built, Scrum master, etc

What practice as the best to support traceability do you think?	02 practices are ”create knowledge and Amplify learning” because you have to set the whole work, how you do things. These 02 are BEST for Traceability because you have to learn how you work, how the whole sys development is done or should be done, a lot of parts in this. My focus is now to implement in code sys development is standard: ISO/IEC 15288. So sys development, then you have thick ideas how you should do sys development and in the dimensions reds handling how this fits goals, other things to do in projects.
Do you think traditional tools can be used in Agile?	I think traditional tool still work for Agile cause you keep track of customer needs and how customer requirements are broken down to subsystems, you have to understand how sys broken down. You have traceability in the Refinement, that is independent if you work in Agile projects or if you work in Traditional: I mean break down is the same either with Agile or Traditional projects. My opinion is you have small projects, you are Agile and customers are happier with sys. You not need break down to subsystem, sys is very small. Then you might not need the tools because you work in group and deliver small project to customer. If you have big sys, you need break down complexity as you cannot have 200 pp working in Agile in one sys. You have to break to subsystems, sys elements. You work Agile with subsystems, you have several Agile teams. For Trace, you have Agile teams up to sys level Trace. But Agile team work on small part. So my conclusion: traditional tools may still be used in Agile organizations because you have Agile teams within the whole team.
What do you expect from the future traceability tool?	I think version now with dimension RM is quite good for Traceability and the features we are working mostly is nice print out from tools. You can read DB in nice way.
What benefits of doing traceability?	I think most important to ensure customers satisfaction by doing Traceability.
How about the challenges of doing traceability?	I think to understand the where you have put diff information is challenging. If you have to trace between requirements, but you have to have design docs. You have to put in differ types of information in diff places. You express what is the actual needs from customer and transfer to functions, so that every members know where information should be put, how you express things in diff docs, artifacts in DB cause it is quite confusing if you try add info in wrong places. So important is find information for tracing, especially for new members.
How do you solve these difficulties?	You should think how you put information; you have to consider what main requirements to fulfill customers are. If you balance the number of requirements and size of docs to be traced, and how you refine the diff level of sys, I think developers will see how they can update links and see how benefit they can earn.We have good developers and they also document, and do Traceability. I think it is company culture. By introducing simple tools or simple ways of handling our requirements, e.g. it is easy for developers to see Traceability actually benefits. You need to define the definition of done at the beginning of the project, I think members would have no resistance.
Who should do traceability activities?	Sys or subsystems engineers will do Traceability, we have decided for differ responsibilities. Some people either write sub-system requirements and also makes traceability in tools. At least one person will do, but all engineers have possibility to see the Traceability cause it is part of inform you have to, or parts of system description you understand sys and see how requirements are linked each other.
Did you have formal discussion about traceability? And will you have one in the future?	Yes, we had discussion about traceability before I started working here. We not had formal seminars but we discussed traceability as requirements of work for how we trace things. I think when we implement simple solutions for handling requirements; we have to spread information how to use tools. There will be information meeting in the future.

11.2.9 Company I

Table 11.9: The transcript of company I

Question	Answer
Can you say about your company in general?	We have two offices. One site in Goteborg and the other Guangzhou China and 50employees for each site. Among them are 35 developers for each site. I work for Company 2001. I work with product, requirements for 13 years, I worked with Agile 2008. Every 6 months for a project, 6 teams and around 10 people for a project.
What are important types of traceability?	Traceability mentions a lot of things, I mean we have these tools as main source of information; we have requirements and have them broken down to tasks. An user story is telling what features delivering. Once they are implemented, we have product released. We would not go back to check requirements and do any Traceability. Feature may be slightly changed it is hard, time consuming to go back to change everything and change requirements to reflect what was the outcome. So we do not have perfect or track every single section in red tools. But what we get out is product and docs and docs both technical, marketing docs. We also have system architecture description and part of definition of done per user story is to update this doc rather than update requirements. System architecture description reflects what has been actually implemented (code). By accepting user story, we have ... we demonstrate user story, it is documented. But it is not back track. Something it is changed with an user story, a developer updates his code. When he does something in sys architecture description. That is the fact that is how it works. If there is change with user story, developers and product owners discuss and decide. When there is exception, we accept user stories, it should be written down and we have acceptance criteria per user stories. There is quite often change with an user story.
What tool are you using to achieve traceability?	A member would have 2-3user stories per Sprint. If there is change, product owner and developers discuss. They are very close to talk, discuss. Product owner participates any stand-up meeting with team members. We are using much communication to keep track of change. An user story starts out is very vague, working to make user stories more clear. Then, team members start working stories, if there is a change, it is communication among the team. Always about communication. This tool is for handling requirements and user stories This tool called AGILO tool. We are mainly manual as the size of the company is quite small. So we only need AGILO to support for handling requirements. The rest is communication.
What tools are not USED?	MARS, ReqPro, I heard some other tools but not remember. There are pros and cons to select tools. It depends on some factors such as: resources, projects, etc. I assume no perfect tool.
Does your tool work well in Agile projects?	My tool works well enough, something is quite slow sometime to work with Agilo. Not all tasks are demonstrated here in this tool. But it is enough for us.
Can you suggest an Agile practice to support traceability?	I think TDD is pretty good. You always have tests updated and able to run test continuously. That is good practice. We sometimes do this and we have team members, who are good with TDD.
Do you think mechanisms designed for traditional software can be re-used in Agile?	Every organization needs to adapt one way or another to make shortcuts. So they need to mix, combine whatever they think effectively, e.g. mixing Agile with traditional software development. In 2001, we not have any tool, we mainly use manual like MS office to handle. So I do not know if traditional tools can work for Agile or not.
Can you say about benefits of doing traceability?	I think it is the way to improve the way of working and cooperation between team members. We have Scrum board (physical stuff, sticker that they move around the board). Improve communication between teams sitting different locations.

How about difficulties of doing traceability?	We try to minimize Traceability by working in pair, like pair programming. The whole team should be involved to work with user stories. So they should not get tasks for individuals, it is the team work. I think it is very important. We have that the issue that developers keep information in their head, not written. I think every company having similar difficulties. It is just personal perspective to resistance, not the whole team.
Who should be responsible for doing traceability?	Everyone should participate in. Members are using AGILO to keep track of change. We have technical staffs to write, summarize documents. Not for developers to focus. In each team, we have technical writer, who collects from team members to write documents. However the system architecture description is very technical. So all members participate in writing. We have configuration role. She keeps track of all part of software, she not using AGILO. But we have that role in our team.
COLOSING	I did not know if we had discussion with Traceability before. Yes, that would be interesting to have a presentation discussing about Traceability in Agile. I might help you to present to my team.

11.2.10 Company J

Table 11.10: The transcript of company J

Question	**Answer**
Opening: general information about company?	72000 employees around the globe. Not exactly the number developers as it is global company may be 20-30.000 developers. As company work all different lines, but I work with Utilities (Electricity, Water). My project is about 50-60 developers. It is big system, cannot estimate line of code or number of requirements. Currently, I am working in project is 05 years. It is worth a couple of hundred millions SEK.
What are important types of traceability?	Probably requirements, involving product owner because that is very important. Product owner or customers need involve much in work for tracing from stakeholders to requirements. Various stakeholders contribute. Obviously, customers need contribute to requirements. For design, it is architects, developers or both, they need involve in that. Actual code, if you do unit test, should be tester and developer, coder involve. Others artifacts involve documents delivers and could be users of subsystem, other people involve a lot of stakeholders.
What factors are important to be considered to add traceability?	Yes, we set conditions; we are working on Billing systems. We consider all types, we know what suppose to do, when time to do, so on. It is matter of Balance, right? what need to do, we need most. If our work is poor now, we have to fix later. So, we need have Traceability for most projects.
What traceability tools are you using?	We have RepPro Repository (all source code - person handle), Issues Tracking System (all work items, you know what been doing on long road, use cases, test case). It is fairly traceable, using MS Virtual Studio, team foundation server are main tools
What benefits of tools?	These tools are very beneficial to my organization. We use initial tracking system, where we put requirements in, we create issues, we break requirements into work items, and they are contained in tools. Everything you work with (documents, code) is tight to these issues.
What tools are NOT used?	We had used Scrum Work before, e.g. Scrum Board in Agile. But it is not pretty good. So we used other tools instead. They do not work for us, in a small project they may work. But we are too big projects, thus we need tools to control our traceability tasks.

How do your mechanisms work well in Agile?	Yes, tools work fairly well with Agile practices I would say so. I mean you still use Excel sheet, white board also, go off line sometimes. Pair-programming is not, TDD is very good as you got actual artifacts or needs, docs, code, that help you have traceable system.
Tools working for traditional software can be RE-USED in Agile?	Yes, we actually used RUP, Lineal as drafter for Agile. We work in short situations, but preparation for SRS, so on. We used Lineal as it is impossible to coordinate what the system need.... as the system is too big. If you have 2-3 months with 6-7 people, it is fine to work only with Agile. If 5 years project with 70 people involving, you need have more formal methods to do these tasks. Yes, we need to mix software development methods. To sum up: yes, tools for traditional traceability can still be used in Agile I think so, but you need also augment tools with something else as well probably.
What do you want with future tools?	It depends on types of Traceability you want to achieve, but I think you need something that connect everything from requirements to code, you need to traceable lines, you need to break requirements into items, work packages and connect code with test case, whatever docs you need to be traced. So you need to break down requirements to see what will all end up with different components.
What benefits of conducting Traceability?	If you do not add Traceability, you will be trouble sometime. Traceability will have all these things as you know something is broken down, you need able to trace. I agree with all purposes of traceability in HANDOUT-02. It makes easier to work when you know where to find information because all work related to information. Right? If we deliver products not working, we must fix this. Thus, everybody here is concerned about Traceability. Traceability makes easier to explain how things have been done. If you have artifacts, they help you out with these.
What difficulties of adding Traces to Agile?	You need to have traceability; people often search Agile as they do not want to make documents, so on. But it is not right way to do that. Agile still need documents, saying what have done. You need to be able to get understandings what happening. I think some are lazy as they do not want to do their work. They think less document mean NO document. There is big difference here about the understanding less docs. Even, some people have resistance to making docs and updating traceability links.
How you cope with difficulties?	They think more fun to write code, but you need to take care of your own mess up. You need to motivate the team make documents, update test cases, whatever, as they make your life easier afterward. It is not issue as it needs to be done. I mean, but the definition of done, you need to say that what have done. People need to fulfill what they commit to do. You need to be aware of what importance for project or product, but not personal importance. They need to do, it is not option. It requires them to deliver documents, I do not care who make, and someone has to make documents. It is not their choice, it has to be done. As coder, you need also review documents. It not an issue as it need to be done and will be reviewed. Yes, it is compulsory, we agree definition of done. You will not finish until you have done with documents as documents belong to definition of done. We have resistance when starting; I think that people understand why we need to do things. Thus, now we have very little resistance. People now realize why doing traceability as they understand the purposes of Traceability.
Who should do traceability in a team?	Everyone needs to participate in traceability activities. We set up frameworks made by configuration manager and team members need to follow. Configuration manager will not want to tell details what to do to members, who need to do specific things. Also the architect participates in making frameworks.
Closing: if you have previous seminar on Traceability? And in future?	We had workshop how to work with Agile concluding about taking care of Traceability. We have mixed everything, from V to Agile models. We need to depend on situations, project by project. I work 14 years, I use Agile for 06 years, we always say the importance of traceability.

11.2.11 Company K

Table 11.11: The transcript of company K

Question	Answer
How many employees are working for your companies?How many software developers?	I have 6 years of experience as a developer. I worked for 02 companies and work for my current company for about 01 year. The company has about 40 employees and around 30 software developers. We mainly work with customers in Telecoms sector. It is quite difficult to estimate how many requirements per project. About 8-10 people participate in one project, which may last 06-20 months.
What are important types of traceability?	For tracing from requirement to other artifacts, the requirements should satisfy all relation types among requirements (Req->Req) because the requirements should be clear, exact, and detailed enough for Design, implementations (Code) and Test to follow. Otherwise it is easy to design, implement and test different functions for the same set of requirements. The requirements should not overlap or conflict each other. Though, requirements don't depend on the other artifacts. Design, however, depends on requirements to define a set of functionalities that satisfy the requirements and are detailed enough at API (Application Programming Interface) level for developers to implement. It does not instruct how the API to be implemented though. During development life, the design can evolve to one another design to reflect the changes from requirements, it could be overlapped but should not be conflict with other legacy designs to maintain backwards compatibility. On the other hands, Code should depend on requirements and design. For example a function should work as it's defined and must satisfy performance requirement. Hence Code should also evolve all the time to meet the changes from requirements and designs. Code can be overlapped but should not conflict with other code. Testing should depend on requirements and designs. For requirements, testing should make sure the code meet the requirements via functional testings. For design, testing should make sure that the code meet the design via unit testings. Finally, the stakeholders has nothing to do with the software design nor implementations but creating and maintaining the requirements throughout the development life.
What types are not important?	The overlap type is not so importance since it does not block the development process. The others are importance, especially the "satisfaction" because they could block a development eg not backward compatible with previous releases due to conflict.
What mechanisms are you using to obtain traceability?	SCRUM, JIRA and GIT. With SCRUM and JIRA, requirements can be created and tracked from design to code and test. Each requirements and designs are transformed into concrete tasks and the tasks' implementations are tracked using source code version control, git and/or svn.
How do your tools work well in your projects?	So far SCRUM seems to work well with us since we usually have small teams, 5-9 man for different projects. It's simple and can quickly adapt to changes from requirements and designs since the SCRUM's sprints are usually short, 2-4 weeks. On the other hand, git helps manage source code very easily and quick.
Do you think mechanisms designed for traditional software methods, could be used in Agile?	I think that tools do not depend on what software method you are using. They just help you to do work faster and more effective regardless of what development methods you are applying. Thus, Tools can work either in traditional or Agile software development.
What do you expect from a future traceability mechanism?	Simple to use and less time consuming

What benefits do you gain when doing traceability?	To keep the developments satisfy requirements' changes because tasks are prioritized and implemented in short sprints. Since the implementations are changed at small steps, it's easier to maintain the code hence usually maximized the code stability. The implementation is done with simplicity in mind, so it's much easier for different developers to work on the same code. As the result, more stable releases are delivered to customers.
Can you say some challenges of doing traceability?	One difficulty is for R&D or prove-of-concept projects, usually it's hard for developers to estimate the time for each tasks. Because for such tasks, it's usually that developers haven't had much experience with. Hence the sprints are harder to maintain. Another difficulty is to have an experience team in which all team members has fairly similar knowledge level of almost every parts of the project's implementation so that one can switch tasks when needed. So it takes some time for team members to gather necessary knowledge and work together effectively.
What are the main causes of these difficulties?	Basically, experience from team members is the key to handle things.
Some team members say: updating traceability links is waste, why is that? and how to handle these challenges?	Defining "DONE" definition for each teams. For example, for developers, "DONE" for a task is when: - Verify the implementation resolves the task eg. Functional test. - Provide test script (if any) to reproduce/verify if the task is resolved. - Get the implementation reviewed. - Commit the implementation to source code version control. - update task's status in the issue tracking system eg. JIRA.
Who should be main responsible for traceability?	Everyone is doing traceability activities. Project managers and scrum master do traceability activities at requirements and tasks' status as well as sprint's performance. Engineers do traceability activities at implementation level e.g. using source code version control and issue tracking systems.
Closing questions	No, haven't heard about a formal seminar. Only from company's internal education.

11.2.12 Company M

Table 11.12: The transcript of company M

Question	Answer
Can you say some information about yourself and your company?	I have worked 2 years for current company, and total 16 years in different companies. I work as project manager, mainly for software development projects. My work is very close to requirement management.In my office, a big city of Sweden more 200 people. We work with public, government, different sectors since we are the big consultant. The company focuses on key accounts: IKEA, Energy, Sony Ericsson, etc. For small project some months, about 7 developers. For big project about 1,5 years, 15 developers participating.
What are important types of traceability? Why?	Dependency is important. Tracing refinement is usually mandatory; if you do not do that your reqs are mad would not make any sense. Depend on methodology for expressing requirements, you have wordings and put in Agile context. You start out with systems, you will refine them with user story. These two are very basic in Agile. Talking to satisfaction, you usually trace from test back to reqs is important definitely. You will have clear picture for criteria satisfying requirements, so and so, it is so important. Contribute is one I also pay attention to.

What types are NOT important? Why?	Evolution (hmn...) I would not think as important and put too much effort on this type. Evolution is not import because it is useful to go back to reqs as stances, but I think we reply more ad-hoc for maintenance or whatever. When they get questions, you have issues related to functions, they will probably lean what they know about system. That is way we work. Overlap is not important or I put little or no effort on this type. May be I am not good example. Conflict is not import because I not expect information. I would reply on BA or whoever working on reqs, they should keep track of that. Rationale is not quite sure to me for me to understand.
Do you set conditions before you decide the level of traceability?	Oh yeah, there are many factors. I would say it depends on matrix we are using to develop products. I do both for Agile and not Agile projects. My previous with IKEA, big project. It is a non Agile project, we had the model for this project, which determine what kinds of traceability you need. So I think the model of project will decide level of traceability. Depend on products, which decide add the level of traceability: e.g. security product requires details.
What tools are you using to achieve traceability?	Now, we use excel sheet for some extends. Then you put in systems such as: MINGO, Clap, JIRA Enterprise sold package, Repository. We use IBM rational for big projects. We have ReqPro, Clear Quest for tracking requirements. I used TRAC, Wiki for several projects before. These tools are good and simple. We used Studio, Tear Fet version in some projects.
These tools are beneficial for your company?	Some tools are beneficial, depend on projects. I talked about MINGO. I think IBM rational is good. Team foundation (forth ....) is OK for use.
How to use these tools?	You start with words, taking notes, and rapidly move lists into Excel. Those will be actually statement, goal, rationale. Then having requirements of product and write requirement statement in whatever format. That would also be done in Excel, if you use tool may be that way, the project is large. Typically, you use Excel; you have sort of actual development projects. You enter all requirements into systems, then you use during the development. So you combine Excel with tools effectively.
What do you know some tools, but are NOT used? Why?	Well, Reproof, tools in databases for tracking docs. Looking at my organization, I will like use MINGO, but it is expensive. JIRA is also expensive for us. Some tools are mandatory within the organization. So you can not buy more other tools.
These tools work very well in Agile?	The mandatory team foundation server tool. It does not work so well with Agile practices. I would do not recommend not using this tool, but we have to use this tool. But MINGO and Tell Low are working very nicely with Agile practices. Tell Low is very simple, MINGO allows you involve tracing requirements to design, and other artifacts, etc
What Agile practice is the BEST to support traceability?	I would say Acceptance Test cause I think that clear acceptance criteria to requirements, to design, to solution (code) is important. Also automated testing, expressing the acceptance test or criteria in Code. TDD is definitely good for tracing. Typically, I see many teams starting TDD then moving on to automated acceptance tests. So these 02 practices go together. You are kind of move from reqs, accept criteria for requirements, choosing solutions, satisfying those in code. They mix traceability in these steps.
Tools supporting for traditional software development methods can be used for Agile?	So, I think so, many tools in traditional can be used in Agile. But, I think there will be friction, I think it is best to find tools for Agile at certain stance because traditional model you want tools to support different kinds of roles, information, access. They have work flows, different things. Those are opposite with ways you are working in Agile, where you have cross-function team, you amplify roles in Agile. So it better to find tools for Agile. Of course, you can use traditional tools, but they may be painful.

What feature of tool are you looking for future tool?	I have not thought much about future tool. But I want virtual white board, it is very useful because it is overview of I also want .connections between requirements, all kinds of work tracking. Connections between traceability and work and finished products. You have nice center face to see ... It is very easy using white board as the very traditional, natural compared with IBM rational, Clear Quest. Sum up: a nice virtual white board, nice report and progress to trace requirements to finished product.
What are benefits of adding traceability in Agile project?	I think we reach our goals, we set goals, date for finished products. So traceability from estimated requirements, then going on to solutions phase, going on to functionality that satisfies requirements. I will use this matrix to keep track progress of the project. Traceability is supporting for tracking progress.
What are challenges of adding traceability in Agile project?	It is not easy to explain the process to team members to understand how traceability to work for requirements, how develop tasks. Basically adding process of working or breaking down reqs into certain tasks. That is the challenge, it is center that explain the traceability in Agile process. You need business analysts (BA) working requirements, helping customer to express their need... you need developers to understand how to implement traceability at the same time they develop functionalities. So the challenge is the explanation to every member to understand the importance of traceability. Expressing the need of Traceability to developers is challenging BA. You need goal, purposes, etc of traceability to explain to developers.
How you cope with difficulties?	In Agile projects, we remove, reduce bad process, manual work, make things more active. You need select good tools, you need to support developers, teams by getting very good tools. Developers will not find difficult to do traceability. Thus, selecting good, simple tools to make traceability easy is very important.
Who do traceability tasks?	All members need to participate in doing traceability. That is the only way, from my much experience.
Have you heard a formal discussion about traceability? Is it helpful to have such a discussion in future?	I have not heard that we had formal discussion. But I think that it may be helpful.
Do you want to add up something?	One observation: achieving high level of traceability is very cheap. It depends on projects. Sometimes, traceability is seen as compulsory. Some projects, it is not. We cannot go back to see traceability because of the delivery time.