December 2013

Missing Requirements Information and its Impact on Software Architectures: A Case Study

Md Rounok Salehin
The University of Western Ontario

Supervisor
Dr. Nazim Madhavji
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

Missing Requirements Information and its Impact on Software Architectures: A Case Study

(Thesis format: Monograph)

by

Md Rounok Salehin

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract:

[Context & motivation] In the development of large, software-intensive systems, the system's requirements are seldom, if ever, concluded upon prior to commencing with systems architecting. Research shows that, in order to manage development and domain complexities, instances of requirements engineering (RE) and systems architecting (SA) processes tend to inter-weave. **[Question/problem]** However, missing requirements information can cause one to create (or recreate) the needed information during different SA activities. While backtracking in the software development process is known to be costly, the costs associated with missing requirements in the SA process have not been investigated empirically. **[Principal ideas/results]** We thus conducted a case study where we investigated to what extent requirements or requirements attributes' information found missing during the SA process and impact of that missing information on SA in terms of effort. The study involved five architecting teams that involve final year undergraduate and graduate students enrolled in the university course on SA, working on architecting a system falling under the "banking" domain. Our result shows that, architects did find requirements and requirements attributes' information missing while architecting. Among requirements information, architects found that, *system functionality information*, *constraints information* and *system interaction (users/systems) information* are missing in requirements at higher percentages. Within requirements' attributes, architects found requirements priority, *dependency* and *rationale* missing at higher percentages. It is also found that, out of total time spent on architecting the system, effort given to recreate missing requirements information is higher for group3 (21.5%), group1 (18%), and group2 (17%) than group4 (12.37%) and group5(10.18%). **[Contribution]** The anticipated benefits of the findings are, it can motivate researchers to venture into other areas of software engineering (such as coding, testing, maintenance, etc.) from the view point of missing requirements information and its impact on those areas. This knowledge could

help software practitioners to decide what kind of information needs to be taken care of, during RE process, that could possibly ease SA process and later development phases. To the best of my knowledge, this is the first work which focuses on, to what extent requirements and requirements' attributes information found missing during SA; characteristics and impact of those requirements missing information on SA process in terms of effort.

Keywords: software engineering, requirements engineering, software architecting, requirements attributes, requirements document, missing information, empirical study.

Acknowledgements

First and foremost, I would like to thank Dr. Nazim Madhavji for his constant guidance and supervision. Without his strong motivation, valuable inspiration and support, this work could not have been completed. Your passion for education and research are admirable.

I would like to thank all the members in my research team. Special thanks go to Md. Rashed Iqbal Nekvi who has suggested many valuable ideas for conducting the research.

I would like to thank Muhammad Iftekher Chowdhury and Dr. Remo N. Ferrari as there research works helped me a lot with diverse ideas to conduct my research.

I would also like to thank Dr. Shahedul Islam and Rafiqul Islam for giving suggestions on statistical analysis with their diverse knowledge in Statistics.

I would like to thank the participants of the study; I greatly appreciate the effort that was expended in the research, and hope that the research output will be of good value to everyone.

Last, but not the least, I would like to thank my parents back home, Md Mosharraf Hossain, Mrs. Sayma Hossain and my little sister Shoria Sharmin for their continuous source of support. I am grateful for their love, patience and trust that acted as a driving force for the completion of the work.

Glossary of terms and definitions

Requirement:

A requirement is a statement of what a system is required to do and the constraints under which it is required to operate.

Requirements document:

The requirements document is the official collection of all the requirements of a system for different stakeholders (e.g., customers, end users, developers etc.).

Software architecture:

Architecture is a description of system structures. It is the first artifact that can be analyzed to identify how well quality attributes are achieved, and it also acts as the blueprint of the project.

Requirement attributes:

It refers to different properties of requirement that describes definition, importance of the requirement, and also describes impact of the requirement on overall system development process. In our thesis we have considered following requirement attributes:

Source of requirement: It refers to the origin of a requirement where it is elicited from.

Priority: It refers to the relative importance of one requirement to others.

Dependency: It refers to the set of requirements on which a given requirement is dependent.

Rationale: Rationale captures the information about the intent behind a requirement.

Assumption: It refers to any proposition regarding a requirement, which has been considered to be true during specification of the requirement.

Missing requirements:

It refers to requirements information that is not found in requirements from architects' point of view. We have considered following information in requirements:

Functionality information: It describes the behaviour and functionalities of the system i.e., how a system should function against different set of inputs (i.e., technical details, data manipulation and processing etc.).

Data information: It describes how the system deals with user's information or system information.

Constraint information: It describes information relevant to technical aspects of the system (i.e., capacity, speed, memory etc.).

System interaction (users/systems) information: It describes information about how the system should interact with different users and other systems.

Missing requirements attributes:

It refers to requirements attributes information that is not found in requirements, from architects' point of view. We have considered all requirements attributes as described above.

Architecturally significant requirement: Architecturally significant requirements are a subset of the requirements that need to be satisfied before the architecture can be considered "stable". Typically, these are requirements that are technically challenging, technically constraining, or central to the system's purpose.

Crosscutting requirement: It refers to requirement that depends on one or more requirements and cannot be satisfied until other requirements are taken into account.

Quality requirement: It refers to requirement (non-functional) that deals with *"Quality Characteristics"* of the system (i.e., performance, usability, reliability etc.)

List of Contents

List of Tables

List of Figures

List of Abbreviations

SE: Software engineering

RE: Requirements engineering

SA: Software architecture

SRS: Software requirements specification

ASR: Architecturally significant requirement

GQM: Goal, question and metric

QR: Quality requirement

ADD: Attribute driven design

Appendix

- Instruments design (Templates questionnaires)

Chapter 1: Introduction

1.1 Problem description

Requirements engineering (RE) and software architecting (SA) are two front end activities in the Software development lifecycle which greatly impact the overall success of a software project. The flow of *"authoritative information"* and control in requirements within RE and SA are maintained through software requirements specification (SRS) [25]. SRS plays an important role in software development projects. SRS contains all relevant and *"official"* information that is required to implement a software system. It is also considered as one of the *"main sources"* of information for software architects [25]. To achieve a successful design approach of any system, specifications of quality attribute requirements are very important and there should be a linkage between requirements specification and design approaches [5]. SA activities (i.e., defining quality attribute scenarios, tactics, patterns, views, etc.) while architecting any system, depend *"heavily"* on requirements and mostly on quality attribute requirements [38]. There is a systematic relationship between general scenarios, concrete scenarios, architectural tactics, and design fragments while architecting a system [38]. Besides, tactics are also dependent on the relationship between design decisions and quality attribute requirements [5][3] and it is an oversimplification, that architecture can be derived from requirements in a single instance [10]. On other note, there is one to many relations between requirements to design [24] and the translation from requirement to design sometimes results in misconception due to the lack of information [24], that could be requirements or requirements attributes' information as well. This requirements' missing information found while architecting could force architects to recreate those information, which would require additional effort on their part. Even though textbooks (e.g., [5], [36], etc.) and standards (i.e., [31], [32]) suggest recording all the artefacts and necessary related information as part of the SRS, it is impossible in reality because keeping track of all the information goes beyond the ability of the temporal nature of

human mind in acquiring knowledge. These requirements' missing information cause problems in different software development phases in the form of errors, misunderstandings and parallel work [51] and affect the actual product in the form of architectural drifts and incorrect design [24] leading to software defects [23][8]. Standish Group report [57] found a striking 28% of projects were cancelled completely. Top factors of failure were lack of a clear statement of requirements in SRS, and incomplete and changing requirements. In a Siemens project the root cause analysis done by Siemens Corporate Research(SCR) showed that 40% of the defects were caused by incomplete or not at all recorded requirements information in SRS [23] [28].

Even though requirements missing information affects other software development phases like product quality [28], product management [28][14][8], software maintenance [28][14] and software quality [42][9], for time and resource constraints our work focuses on finding the extent to which requirements and requirements' attributes are found missing during SA process and its impact in terms of effort on requirements while architecting a system.

1.2 Focus of research and motivation

Research shows evidence about the impact of missing requirements' information on architectural design decisions [25] [39]. Requirements to design transition is defined as the most severe among different software information leaks between development phases [24]. Relative expense for fixing defects, introduced in requirements, are three times higher if found during architecture phase and five to ten times higher if found during construction phase of development of the system [27]. But we could not find any empirical evidence of the extent of missing requirements and requirements' attributes and its impact on SA, in terms of effort. This motivates us to look into the extent of missing requirements and requirements' attributes' information during SA and its impact on SA. Our research is focused on identifying the extent to which requirements and requirements' attributes found missing during SA and its impact on SA process in terms

of effort. Our research also focused to identify the extent of different types of changes made to requirements and requirements' attributes, and different characteristics of changed requirements that are important from architects' points of view.

1.3 Originality

To the best of my knowledge, characterisation of different requirements and requirements' attributes, found missing during SA activities has not been carried out by any other researcher. Even though there is mention of *"information leaks"* in different development phases [24], there is no scientific research that I am aware of that is engaging this study. While there is some evidence in the literature of the knowledge gap between architecture and coding [19] and architects' assumptions on requirements [1], it is not known to what extent, requirements and requirements attributes' information are found missing during SA and how frequently it introduces rework in missing requirements information in terms of effort.

1.4 Research approach

We have used the case study approach as it allows for an empirical investigation of a contemporary phenomenon within its natural context using multiple sources of evidence [60]. The type of study conducted was a *multiple-case* study design [15] where we had five parallel cases (i.e., one case per architecting team). Despite this, it was an *exploratory* study in that we had no initial hypothesis, and we did not know which phenomena were important (i.e., impact of missing information on architecture in terms of effort). This is because, to our knowledge, there was not much background literature on related research questions. Details of the study design are described in section 3.1 and section 3.2.

1.5 Research contributions

From our findings we found that, *constraints information* and *system functionality information* are found missing in requirements at higher numbers for different groups. Among different missing requirements' attributes, *priority*, *dependency* and *rationale* of requirements, have higher percentages for different groups. Our findings also state the extent of different changes made to requirements and requirements' attributes. Among them modification of requirements and requirements' attributes are having higher percentages for all the groups. Our results also state whether the changed requirements have certain characteristics (i.e., architecturally significant requirement, crosscutting requirement, and quality requirement) that are important from architecture's point of view. Our findings also show the evidence about the impact of missing information on SA, in terms of effort. Among them group 3 spent more time on missing information; that is, 21.5% of total time was spent on the project. Details of our findings are described in Chapter 4.

1.6 Implications of the findings

To the best of my knowledge, there is no empirical evidence that shows, to what extent requirements and requirements attributes' information is found missing and impact of this missing information on SA, in terms of effort. This work could motivate the practitioners to use RE process in a way so that minimal requirements' and requirements attributes' information goes missing. This work could also motivate researchers to venture into other areas of software engineering (i.e., coding, testing, maintenance, etc.) to find out the impact of requirements missing information on different development phases. For example, knowing what kind of requirements and requirements attributes' information are missing and how much it is affecting SA in terms of effort could help

software practitioners to decide which medium or approach to avoid and to use with caution in RE and SA processes.

There are already some research tools (e.g., [23] and EGRET [51]) and tools used in industry (e.g., Rational Team Concert [46]), which provide traceability between software artefacts and communication artefacts (e.g., meeting videos, email, chat, etc.). Researchers could improve features of different tools or try to develop new tools if needed by keeping our findings in mind so information goes missing as infrequently possible. Details of implications of our findings are described in chapter 5.

1.7 Thesis organization

Chapter 2 discusses the relevant background literature and research gap. Chapter 3 explains the research questions, metrics, experiment design and threats to validity. Chapter 4 presents the data analysis, results and interpretations. Chapter 5 explores the implications of the findings and Chapter 6 closes this thesis with limitations of this study, conclusions, and future work of this research.

Chapter 2: Literatures Review

The review of the literature culminated into broad areas discussed in Section 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, and 2.8. Section 2.1 describes about the RE process and Section 2.2 shows the commonly used architecture structure. Section 2.3 describes what is meant by requirements document in this thesis and Section 2.4 describes the attribute driven design (ADD) approach that is used by architects in this thesis for architecting the system. Section 2.5 describes, why in current practices, requirements' information go missing in RE documents. Section 2.6 discusses key empirical research on importance of software documentation, knowledge gap between development phases, requirements problems and its impact on SA, and impact of missing/incomplete requirements in industries. Section 2.7 describes importance of requirements characteristics in SA. Section 2.8 describes importance of requirements' attributes. Section 2.9 describes the overall research gap based on sections 2.5, 2.6, 2.7, and 2.8.

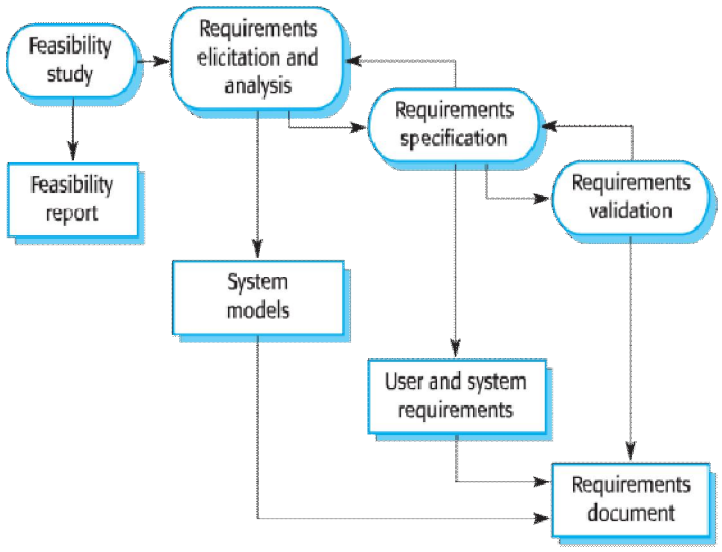
2.1 Requirements Engineering (RE) process [36]

There are different RE processes that vary based on application domain, people involved and the organization developing the requirements. The common RE process is given in Figure 1. Generic activities that are common to all processes are:

- **Requirements elicitation:** It involves the task of working with customers to find out about the application domain, different services that the system should provide and system's operational constraints. It can also involve different stakeholders (i.e., technical staff, end-users, managers, engineers etc.). It is also named as *"requirements discovery"*.

- **Requirements analysis:** It involves the tasks that should be implied on elicited requirements.
    - Requirements classification and organization: it refers to group related requirements and organizing them accordingly.
    - Prioritization and negotiation: Prioritizing requirements and resolving requirements conflicts.
    - Requirements documentation: All requirements are documented and input into the next round of the process.
- **Requirements validation:** It involves the task of identifying whether the requirements define the system that the customer really wants. It checks for different requirements concerns (i.e., validity, consistency, completeness, realism etc.).
- **Requirements management:** It involves the task of managing existing requirements and changing requirements during the RE process and system development.

Figure 1: The common RE process [36]

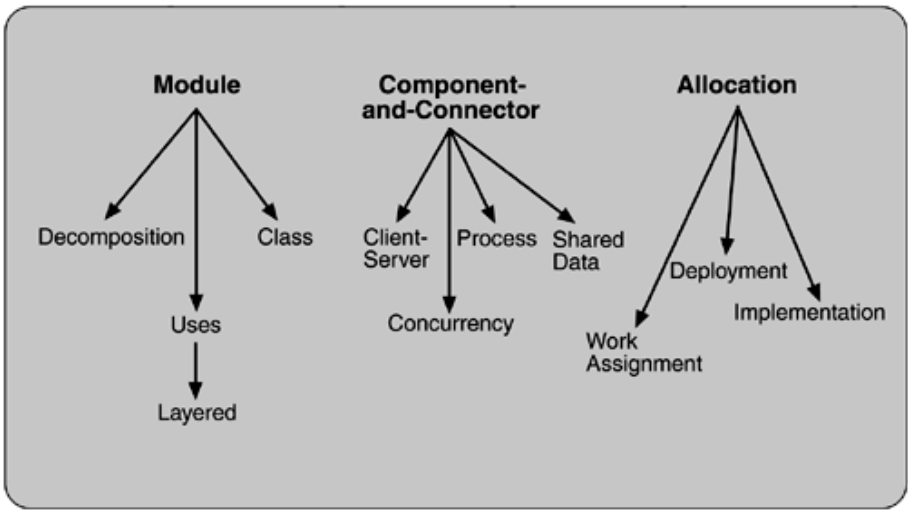


2.2 Software architecture [5]

Architecture is a description of system structures. It is known as the first artifact that can be analyzed to identify how well quality attributes are achieved, and it also acts as the blueprint of the project. It serves as the communication vehicle that can be transferred to new systems. Common software architecture structure will be most likely as given in Figure 2. As architecture acts as a backbone of the project's design, it must be communicated clearly without any ambiguity to all of the stakeholders (i.e., customer, user, project manager, coder, tester, etc.). All stakeholders need to understand it in a better way as:

- Developers must understand the work architecture requires of them.
- Testers must understand the task structure it imposes on them.
- Management must understand the scheduling implications that the architecture suggests, and so forth.
- Towards the end the architecture documentation should be informative, unambiguous, and readable by many people of various backgrounds.

Figure 2: Common software architecture structure [5]



2.3 What is a requirements document

A requirement is a statement of what a system is required to do and the constraints under which it is required to operate. The requirements document is the official collection of all the requirements of a system for different stakeholders (e.g., customers, end users, developers etc.). A requirements document is also known as 'functional specification', 'requirements definition', 'software requirements specification (SRS)' etc [36]. Requirements can be written in different formats like natural language description [36], formal specification [36], user story [2] etc. There are various tools varying from simple word files [4] to complex multitier data base management systems [22] are also used for storing requirements documents. In this thesis, we refer to requirements document using the terminology 'requirements document' and "SRS" respectively.

2.4 Attribute Driven Design (ADD) [5]

ADD is a design approach to define software architecture. It is based on the decomposition process on the quality attributes the software has to fulfill. It is a recursive decomposition process where, at each stage, tactics and architectural patterns are chosen to satisfy a set of quality scenarios and then functionality is allocated to instantiate the module types provided by the pattern. In our thesis, for architecting the system, architects have used this approach.

ADD steps [5]:

- First, architects need to choose a module to decompose. The module to start with is usually the whole system. All required inputs for this module should be available (constraints, functional requirements, quality requirements).

Then they need to refine the module according to these steps:

- Choosing architectural drivers from the set of concrete quality scenarios and functional requirements. This step determines what is important for this decomposition.

- Choosing an architectural pattern that satisfies the architectural drivers. Create (or select) the pattern based on the tactics that can be used to achieve the drivers. It helps to identify child modules required to implement the tactics.

- Instantiate the modules and allocate functionality from the use cases and represent using multiple views.

- Define interfaces of the child modules. The decomposition provides modules and constraints on the types of module interactions.

- Architects need to verify and refine the use cases and quality scenarios and make them constraints for the child modules. This step verifies that nothing important was forgotten and prepares the child modules for further decomposition or implementation.

They need to repeat the steps above for every module that needs further decomposition.

2.5 Why requirements information goes missing in RE documents

Different types of communication channels (e.g., face to face, e-mail, chat, etc.) are used in RE. Because of their highly communication intensive nature all types of information are not possible to document. Besides, requirements document is considered as the "*main source"* of information for later development phases [25]. Here are some of the reasons why requirements or attributes of requirements could go missing during RE:

- In current documentation practice one of the stakeholders present in the meeting takes the role of the scribe [53]. It is seen that sometimes different stakeholders involved in the process may not share a common language or project knowledge, and due to that the notes of the scribe can be incomplete, inconsistent or incorrect [53]. For example, the scribe may misinterpret a statement, note something incorrectly or partially, or omit important statements made by a stakeholder [53]. This might cause missing information from the overall documentation.

- Sometimes important information is discussed outside formal documentation [51] [12]. Fluid information (such as meetings and oral communications, blogs, chats, informal wikis, and phone calls, etc.) usually get ignored from explicit documentation [51].

- Documentation is effort consuming [51] but quickly becomes outdated [12] [16]. Often it takes few weeks to months for the documentation to be updated [40]. This delay in update might result in information missing in documents.

- Sometimes there is lack of integration between software artifacts and communication environment (e.g., e-mail client, instant messenger, video

conferencing tools, etc.), information gets fragmented across several media (e.g., documentation tools or databases, bug repository, e-mail, text chats, etc.) which leads to frequent context switching (e.g., rechecking old email, chat sessions, meeting minutes) during work and results in a lack of common understanding and awareness [51]. Important domain information is personalized (i.e., "at best retained in the team member's mind") and gets missing when the team member leaves the project [51].

- Sometime, requirements engineers who are inadequately trained, have inadequate access to stakeholders and other sources of the requirements. They are given inadequate resources or authority to properly engineer the requirements [21].

2.6 Empirical research on:

We have discussed some key research here but it is important to note that the actual research focus for some of the studies mentioned below spans a broader area. Because of our scope, we only discuss issues and findings that are related to the importance of software documentation, knowledge gap between development phases, requirements problems and its impact on SA, the importance of different characteristics and attributes of requirements in SA, and the impact of missing/incomplete requirements in industries.

2.6.1 Importance of software documentation

Lethbridge et al. reported three studies on the use of software documentation [40]. For their study they have used methods like interviews, surveys and observing individuals' work. Their study showed that documentation other than testing and quality documentation (such as test cases and plans) are rarely updated. Even if the changes are

made to the documentation it usually takes several weeks for the documents to reflect actual system changes. They also found that the out-dated documentation might remain useful in some cases, particularly if the high level abstractions remain valid. They also mentioned the necessity of simple and powerful documentation tools and formats.

Beecham et al. conducted a study in 12 software companies ranging from CMM level 1 to 4 to find their software process improvement problems [6]. In their study they divided 200 employees from these companies into 45 groups and used focus group techniques to collect data. Their result shows that documentation issue (i.e., coordination and document management, feedback, post-mortems and data collection) is one of the major problems reported by developers, project managers and senior managers. Amongst the project issues, documentation was ranked number 2 and amongst the top 6 problems identified from all the areas (i.e., organizational issues, project issues and software development lifecycle process), documentation was ranked number 3.

2.6.2 Knowledge gap between software development phases

George et al. described relevant information that is lost during different development phases and termed it as *"Software Information Leaks (SIL)"* [24]. Definition of SIL is given as, "*relevant information once known, but not incorporated in later stages and thereby lost in software evolution"*. They developed a software security impact analysis model and tried to find key dependencies between information. SIL is identified to occur in technology level, domain level, requirement level, architectural level, design level and code level. Improper documentation is mentioned as one of the major causes of architectural drift. SIL that occurs during transition of requirements to design is defined to be the most severe leaks of software development.

Feilkas et al. conducted a case study about loss of architectural knowledge in three industrial projects [19]. They answered questions like: i) to what degree is the architectural documentation kept in conformance with the code; and ii) how well does the documentation reflect the intended architecture, by analyzing architecture documentation,

and by interviewing developers. It is found that, between 9% and 19% of all dependencies contained in implementation, could not be identified in architecture documentation, and between 72% and 90% of the differences between documented and implemented architecture is due to lack of proper documentation information. Lessons learnt in this study are: architectural knowledge gets lost due to the gap between architectural specification and implementation; significant amount of effort is needed for reverse engineering the architecting process.

Albayrak et al. conducted an empirical study with 251 software engineers from eight companies, and 39 projects to show how software engineers responded to an incomplete requirement and how it is having impact on the number of explicit assumption they made [1]. They have suggested as per their findings that, implicit assumptions should be avoided and explicit assumptions should be preferred so engineers do know explicitly that which gap to fill and how to fill. They have shown by using statistical ANOVA method that, preferred response of engineers to given incomplete requirements have impact on the number of explicit assumptions they made, and on average, non-computer-background engineers made more explicit assumptions other then computer background graduates.

2.6.3 Requirements problems and its impact of on SA

Gross and Doerr described the results of explorative studies conducted on two academic practical courses having a total of 13 students [25]. They discussed about architects information need that should be fulfilled in SRS. SRS documents are mentioned as *"official"* information that is required for implementing the system and considered as main sources of information for architects. It also described about negative impacts of over-documented and less-documented SRSs on architects, that in the worst case sometimes they could neglect or even ignore the SRSs. They have investigated to find best-practice artifacts types and their representations from the viewpoint of software architects. Results show that, artifact types, documenting information about system

responsibilities, data, system functionalities, interactions, and technical constraints are found very important from architects' points of view.

Lee and Rine described a methodology named "*Proxy viewpoints Model-based Requirements Discovery (PVRD)"* that provides a framework to construct a viewpoints model from requirements and supports requirements discovery for efficient management [39]. They discussed incomplete SRS with "*missing, not available and hard-to-locate requirements"* and mentioned manual discovery of missing requirements and relationships between them as a "*highly labor intensive"* task. They have applied their model to study a finance application system domain and showed the process for identifying missing requirements and the relationship chain between requirements.

Gumuskaya described the core issues that are affecting the architecture in enterprise projects where complex business, management and technical problems exist [27]. He stated that, requirements information problems in documents have more potential to cause longer system defects and found them to be more expensive as well. As per the results, it is found that, the relative expense for fixing defects, introduced in requirements, are three times higher if found during the architecture phase and five to ten times higher if found during the construction phase of development of the system.

Ferrari and Madhavji described an exploratory case study to determine different types of requirements oriented problems that architects faced while architecting the system [20]. They designed a multiple case study design that includes 16 architecting teams (i.e., one case per architecting team). Architects faced 35% problems due to requirements. Their results also found severe problematic technical areas within requirements oriented problems like, quality satisfaction (22%), requirements understanding (18%). quality drivers determination (15%), abstraction (14%), and modelling quality requirements (scenarios) (12%).

Firesmith has summarized the 12 “worst” of the most common requirements problems from his working experience with real projects as a requirements engineer, consultant, and evaluator [21]. Poor requirements quality in requirements specifications was one of them. He described from practice that, too many requirements in real specifications are found ambiguous, incomplete, inconsistent, incorrect, out-of-date. They do not follow the terminology of the user or business/application domain. He also specified that, in practice, architecturally-significant requirements are accidentally overlooked many times and those are usually non-functional requirements, most commonly quality requirements. This problem happens most of the time because the stakeholders often assume that such requirements are obvious and go without saying.

Kamsties et al. summarized the results of a workshop on requirements engineering held with practitioners from 10 small and medium enterprises (SMEs) [34]. It is found that, they were organized around issues like, product and project characteristics, requirements sources and elicitation activities, requirements engineering products (i.e., the software requirements specification (SRS)), requirements validation activities, and system evolution (i.e., further development activities). As per their findings, SMEs that do produce SRS face subtle problems during the evolution of the system. Among the top problems implicit domain knowledge in SRS is one of them as it is difficult to understand vague requirements for them. Besides, these vague requirements are causing problems in testing as requirements are not traceable enough.

2.6.4 Impact of missing/incomplete requirements in industries

Here is some empirical evidence from existing literature which shows the severity of the problems caused by missing or incomplete requirements in industries:

- A study conducted by the Standish Group [57] found a striking 74% project failure rate, while 28% of projects were cancelled completely and top reasons of failure was lack of user input, lack of a clear statement of requirements in specifications.

- In a Siemens project the root cause analysis done by Siemens Corporate Research(SCR) showed that 40% of the defects were caused by incomplete or not at all recorded requirements in documents[23][8].

- In a study [53], a survey was conducted in 63 software companies in Malaysia. The companies cited problems like incomplete requirements (79.4%), misplaced requirements in a requirements document (37.1%) as some of the reasons behind late delivery of products (76.2%), budget over-runs (58.7%) and poor quality products (44.4%).

- In June 1991 to September 1991, three surveys on 39, 41 and 44 software maintenance professionals (with overlapping participants) were conducted and 19 major problems in software maintenance were identified [17]. Incomplete information in system documentation was ranked number 3 amongst them.

- As per the report [59], industry data suggests that approximately 50% of product defects originate in the requirements. Perhaps 80% of the rework effort on a development project can be traced to requirements defects. These defects are the cause of over 40% of accidents involving safety-critical systems.

2.7 Importance of different characteristics of requirement in SA

2.7.1 Architecturally Significant Requirement (ASR)

In a study [14], the authors described the importance of identifying architecturally significant requirements (ASR) in SRS. They have presented the use of "*Architecturally Savvy Personas (ASP-lite)"*, a method to analyze stakeholders' quality concerns and

validate the architectural design. ASRs are mentioned as the *"driving force"* to architectural design of a software intensive system. It is also used as the selection criteria for deciding between alternate architectural options. It is therefore suggested that, it will be beneficial for a project if we could elicit and analyze these requirements in the early phases and document them as that could help in considering the architectural design of the system.

In early project stages, requirements and architecture are expressed as *"highly informal"* [58] and due to that, often ASRs are overlooked in the beginning of a project [13]. The hints those are uncovering them are sometimes found not well documented as well [29]. In consequence, also the decision knowledge tends to remain implicit sometimes. When requirements and architecture evolve, this informality eventually leads to a loss of decision knowledge. Due to the vague or changing ASRs in documents, sometimes it may not be reflected in the architecture, which results in high costs for later changes [13].

2.7.2 Quality requirement:

Quality (non-functional) requirements (QR) for a system represent a special subset of architecturally significant requirements (ASRs) and describe the non-behavioral constraints on the system [14]. It has the *"biggest"* role in the architectural design [44]. These requirements define the overall qualities and attributes of the resulting system like (i.e., safety, security, usability, reliability, performance etc.). But in practice, sometimes, interdependencies and trade-offs among QRs remain unclear. Sometimes they are not appropriately defined and prioritized in documents. This unusual missing information in documents causes problems during architecting the system and other development activities as well [37].

2.7.3 Crosscutting requirement:

In a study [48], the author defined requirements that crosscut other requirements; known as crosscutting requirements. For example, requirement X crosscuts requirement Y if a software decomposition has been chosen in which Y cannot be satisfied without

taking X into account. So X in this case will be crosscutting requirement. In reality, semantic relationships between requirements could not be identified and they are explicitly recorded in specifications [48]. If some of the relationships among requirements (such as crosscutting influences) remain obscure, developers have the risk of forgetting about them. In the worst case, it is found that those relationships are discovered indirectly by users while requirements are not totally fulfilled by the software after deployment. In both cases increased costs are the result of the requirements deficiencies and it can rise to other troubles such as loss of faith in the developers or even the failure of critical system components [48].

2.8 Importance of requirement attributes information

In a study [41], the authors surveyed the state of practice on rationale management, usage, and challenges of integrating rationale management in RE activities. They stated that, traditional requirements specification only specifies *what* the requirements are, but not *why* these requirements are specified. This causes problem in later phases of development, where requirements get involved due to stakeholders' change of mind when the system is being updated. As a result of their findings, it is defined that rationales in specification should include every decision, its alternative decisions, and underlying arguments leading to that decision (e.g., influence factors, criteria, and negotiations). These decisions also help in prioritizing the requirements as rationale helps to make decisions about what requirements to be implemented first or should be given most resources from architects' points of view [11].

Traceability is another important requirements attribute that defines the ability to follow and describe the life cycle of a requirement, in both forward and backward direction [35]. Traceability has a positive effect on project management (simplifying project estimates), process visibility (accessibility to contextual information about requirements become easier), and maintenance of the system as well.

Assumptions in RE are sometimes found *"too ideal"* and some of them cannot be satisfied in the running system due to unexpected agent behavior [38]. This lack of anticipation results in unrealistic and incomplete requirements that result in poor performance or failures. Improper documenting of assumptions about requirements sometimes results in run time inconsistencies between specification of the system and actual behavior of the system [38]. Sometimes there is *"invalidity"* in assumptions and requirements and that is a source of problems for software developers and users. For an operational system, this problem may imply from a diminishing value the system to a software failure as well [43].

2.9 Analysis of research gap

Studies show (section 2.5) several reasons for why requirements information is found missing during RE includes: (i) use of different communication language [23], (ii) lack of project knowledge between stakeholders [23], (iii) omitting statements made by stakeholders [23], (v) misinterpreting any statement and note something incorrectly or partially [23] and (vi) absence of integration between software artifacts and communication artifacts [51]. But the types of requirements or requirements attributes' information goes missing are not stated within any study.

Studies show evidence of the importance of complete SRS in architecting and other phases [25][9] but rarely discuss anything about what types of information were not there in the SRS or impact of this missing information on architecture or other development phases. In a study [24], the authors describe design decision recording problems that include partial documentation, lack of information relating requirements to rationales, while architecting the system, but did not mention anything about to what extent, requirements were missing in SRS and its impact on architecture and other phases of development.

Research also depicts that, missing requirements information affect software development phases like architecture [24] [28], product quality [53], product management [28] [8], software maintenance [53] [56] and software quality [42] [9]. But

none of them discussed anything about the impact of incomplete information on different phases in terms of effort on this missing information.

In (section 2.7), studies refer to different characteristics of requirements (i.e., ASR [14] [58], quality requirement [14] [44] and crosscutting requirement [48]), that are important from a system development perspective, but none of them really state anything about the extent to which missing requirements or missing requirements' attributes are having this characteristic and how it could affect the architecture or other development. Section 2.8 gives evidence about the importance of different requirements' attributes, but studies did not discuss to what extent they are going missing or its impact on development phases.

In (section 2.6.2), we found a study [24] concerning the occurrence of missing information in different software development phases. The software information leak, at the transition from requirements to design is mentioned as the most severe leak [24] but the study did not mention anything about to what extent, requirements information was missing from requirements to architecture and other development phases. In another study [20] the authors described requirements problems found during architecture but did not state anything about the extent to which requirements or requirements' attributes are found missing during SA or the impact of this missing information on SA, in terms of effort. Studies also have evidence about knowledge loss in architecture and its impact on coding [19], designers implicit assumptions while they cannot avoid getting incomplete SRS when designing the system [1], architects' expectation from SRS [25], but none them explore to what extent requirements information has gone missing or what impact it has on architecture or other development phases in terms of effort and quality.

However, there is a significant research gap in finding to what extent requirements or requirements' attributes are found missing; its impact in terms of effort on this missing information during the software architecting process; and characteristics of changed requirements; impact of this missing information on overall software development life cycle and other development phases (i.e., product management, project management, risk analysis, software quality etc.), that motivate empirical investigation in this area.

Chapter 3: The Empirical Study

This chapter describes the details of the study and the research procedures that were carried out. Section 3.1 defines the overall research goal, specific research questions and associated metrics. Section 3.2 describes the RE document that is given to all the groups and section 3.3 describes the architecting process that all the groups have followed. Section 3.4 describes the research procedures in terms of participants, data collection and data analysis. Section 3.5 describes the threats to validity of our study.

3.1 Goal, Questions and Metrics

We used the *Goal Question Metric* (GQM) [4] to formulate the goal of the study, pertinent research questions and associated metrics to gather appropriate data. In the following, we first define goal of the study, then we describe our research questions and associated metrics.

3.1.1 The Goal

Purpose: To determine and analyze

Issue: Characteristics and extent of requirements and their attributes that are found missing during SA, and impact of these missing information on SA in terms of effort.

Object: Requirements information and requirements attributes information

Viewpoint: From the viewpoint of software architects

Context: In the context of software development project with particular focus on (RE) and (SA)

3.1.2 Research Questions and Associated Metrics

The goal of the study has two dimensions: (i) *characteristics of requirements information;* and (ii) *impact of missing requirements on software architecting efforts*. Our research goal has led us to formulate following research questions.

Q1: To what extent, (i) requirements and (ii) their attributes found missing, for architecting the system?

Q1 can be decomposed into two separate questions i.e., Q.1.1 and Q.1.2 as follows:

> **Q1.1**: To what extent, requirements found missing while architecting.
>
> **Q1.2**: To what extent, requirements attributes found missing while architecting.

Q 1.1: To what extent, requirements, found missing while architecting?

Description:

There are various types of requirements information that could be missed while doing the RE works. To categorize the missing requirements information we used the classification of requirements information as suggested by Gross and Doerr in [25]. In [25], the authors presented the types of requirements information including functional information, data information, constraint information, etc. (see Table 1) that the architects usually expect to be specified in the requirements documents. With reference to Q.1.1, we sought to investigate the extent to which these types of information that were found missing according to the architects. In our thesis we have considered *missing requirements information* (Table 1) by using the term *missing requirements*.

Table 1: Types of requirements information

Types of Requirements information	Description
Functionality information	It describes the behaviour and functionalities of the system i.e., how a system should function against different set of inputs (i.e, technical details, data manipulation and processing etc.)
Data information	It describes how the system deals with users' information or system information.
Constraint information	It describes information relevant to technical aspects of the system (i.e., capacity, speed, memory etc.)
System interaction (users/systems) information	It describes information about how the system should interact with different users and other systems.

From Q1.1, we have one theoretical *"construct"* i.e., extent of requirements found missing while architecting. Thus, we define the following metric (M1) for data gathering (Table 2).

M1: Number of requirements missing, found while architecting the system.

Table 2: Metric M1. – Number of requirements missing, found while architecting the system

Types of requirements information	**Number of missing requirements**	
	Frequency (n)	**Percentage ((n/total)*100)%**
Total		
System functionality information		

System data information		
Constraint information		
System interaction (users/systems) information		
other		

Q1.2 To what extent, requirements attributes found missing while architecting?

Description:

Here, we sought to investigate the extent of the missing information concerning various requirements attributes, which are later discovered by the software architects team. We have used existing literature [4] for the set of requirements attributes. Table 3 depicts the list of requirements attributes (see the 1st column) and associated brief descriptions (see the last column) which have been considered in the study.

Table 3: Description of different types of requirement attributes

Types of requirements attributes	**Description**
Source of requirement	It refers to the origin of a requirement where it is elicited from.
Priority	It refers to the relative importance of one requirement to others.
Dependency	It refers to the set of requirements on which a given requirement is dependent.
Rationale	Rationale captures the information about the intent behind a requirement.
Assumption	It refers to any proposition regarding a requirement, which has been considered to be true during specification of the requirement.

From Q.1.2, we have one theoretical *"construct"* i.e., extent of requirement attributes found missing while architecting. Thus, we define the following metric (M2) and definition of the metric is given in Table 4.

M2: Number of requirements attributes, found missing, for architecting the system.

Table 4: Metric M2- Number of requirements attributes, found missing

Types of requirements attributes	**Number of requirements attributes, found missing**	
	Frequency (n)	**Percentage (n/total)*100%**
Total		
Source of requirement		
Priority		
Dependency		
Rationale		
Assumption		
Other		

Q2: To what extent, changes are required in the requirements and requirement attributes, while architecting the system?

Description:

This question was asked to have information about what kind of changes architects made to the given requirements. We defined three possible types of changes that architects' can do in requirements document.

Add: Adding one or more new requirements.

Delete: Deleting one or more requirements.

Modify[1]: Modifying specific requirement

Q2 can be decomposed into two separate questions i.e., Q.2.1 and Q.2.2 as follows:

Q.2.1: To what extent, changes are required in the requirements?

Q.2.2: To what extent, changes are required in the requirements attributes?

With reference to Q.2.1, we have considered different requirements information described in [25] (see Table 1). There is one theoretical "*construct"*: extent of changes to requirements while architecting. Thus, we define the following metric (M3) (Table 5).

M3: Number of changes on requirements, while architecting.

Table 5: Metric M3- Number of changes on requirements

Requirements information	**Number of changes on requirements, while architecting**					
	Add		**Delete**		**Modify**	
	Frequency (n)	**Percentage (n/(n+total))*100**	**Frequency (n)**	**Percentage (n/total))*100**	**Frequency (n)**	**Percentage (n/total)*100**
System functionality information						
System data information						
Constraint information						
System interaction (users/systems) information						
Total						

[1] In our thesis, we have considered *modification* to requirements or requirements attributes as *missing information* as well. Any individual information of any requirement is itself a requirement.

With reference to Q.2.2, we have considered different requirement attributes described in Table 2. There is one theoretical *"construct":* extent of changes to requirement attributes while architecting. Thus, we define the following metric (M4) (Table 6).

M4: Number of changes on requirement attributes, while architecting

Table 6: Metric M4- Number of changes on requirement attributes

Requirements information	**Number of changes on requirement attributes, while architecting**					
	Add		**Delete**		**Modify**	
	Frequency (n)	**Percentage (n/(n+total))*100**	**Frequency (n)**	**Percentage (n/total))*100**	**Frequency (n)**	**Percentage (n/total)*100**
Source of requirement						
Priority						
Dependency						
Rationale						
Assumption						
Other						
Total						

Q3: What are the characteristics of the changed requirement?

Description:

In Table 7, we have described different requirement characteristics, those are important software architecting [14][48]. Definition of different characteristics of a requirement is given in Table 6. Q3 can be decomposed into three separate questions i.e., Q.3.1, Q.3.2, and Q3.3 as follows:

Q.3.1: How many of the changed requirements are architecturally significant?

Q.3.2: How many of the changed requirements are cross-cutting requirements?

Q.3.3: How many of the changed requirements are quality requirements**?**

Table 7: Definition of different characteristics of a requirement

Requirement characteristics	Description
Architecturally significant requirement (ASR)	Architecturally significant requirements are a subset of the requirements that need to be satisfied before the architecture can be considered "stable". Typically, these are requirements that are technically challenging, technically constraining, or central to the system's purpose [14].
Crosscutting requirement	It refers to requirement that depends on one or more requirements and cannot be satisfied until other requirements are taken into account [40].
Quality requirement	It refers to requirement (non-functional) that deals with "Quality Characteristics" of the system (i.e., performance, usability, reliability etc.) [14]

With reference to Q.3.1, there is one theoretical *"construct":* how many of the changed requirements are architecturally significant. Thus, we define the following metric (M5) (Table 8).

M5: Number of architecturally significant requirements.

Table 8: Metric M5: Number of architecturally significant requirements

Requirement characteristic	**Number of architecturally significant requirements**	
	Frequency(n)	**Percentage ((n/total)*100)%**
Total		
Architecturally significant requirement		

With reference to Q.3.2, there is one theoretical *"construct":* how many of the changed requirements are crosscutting requirements. Thus, we define the following metric (M6)(Table 9).

M6: Number of crosscutting requirements.

Table 9: Metric M6: Number of crosscutting requirement

Requirement characteristic	**Number of crosscutting requirements**	
	Frequency(n)	**Percentage ((n/total)*100)%**
Total		
Crosscutting requirement		

With reference to Q.3.3, there is one theoretical *"construct":* how many of the changed requirements are quality requirements. Thus, we define the following metric (M7)(Table 10).

M7: Number of quality requirements.

Table 10: Metric M7: Number of quality requirements

Requirement characteristic	**Number of crosscutting requirements**	
	Frequency(n)	**Percentage ((n/total)*100)%**
Total		
Quality requirement		

Q4. What is the impact of missing requirements information on SA, in terms of effort**?**

Description:

This question was asked to find out how much effort was spent by different groups to recreate the missing requirements information. We define effort as, how much time architects' spent on recreating the missing information that includes, time to identify the problem and time to address the problem. To calculate effort, we took the time (person-hours.), spent on recreating different missing information.

For individual groups, we compared the effort spent on requirements missing information with total time they spent on architecting the system. We have calculated the total time from time log template (TLT) (see Table 12). It includes the time they have spent on group meetings. There were lag times in group meetings that are specified in TLT. We excluded those times from the total time. Then we did percentage distribution to find out, how much effort they have given on recreating requirements missing information in compare to total time spent on the project.

With reference to Q4, there is one theoretical *"construct":* effort spent on requirements missing information. Thus, we define the following metric (M8) (Table 11).

M8: (Person-hours) spent on requirements missing information, while architecting

Table 11: Metric M8: (Person/hours) spent on requirements missing information, while architecting

Participant group	**(Person/hours) spent on requirements missing information**	
	Total time spent on the project (person/hours)	**Total time spent on recreating requirements missing information (person/hours)**

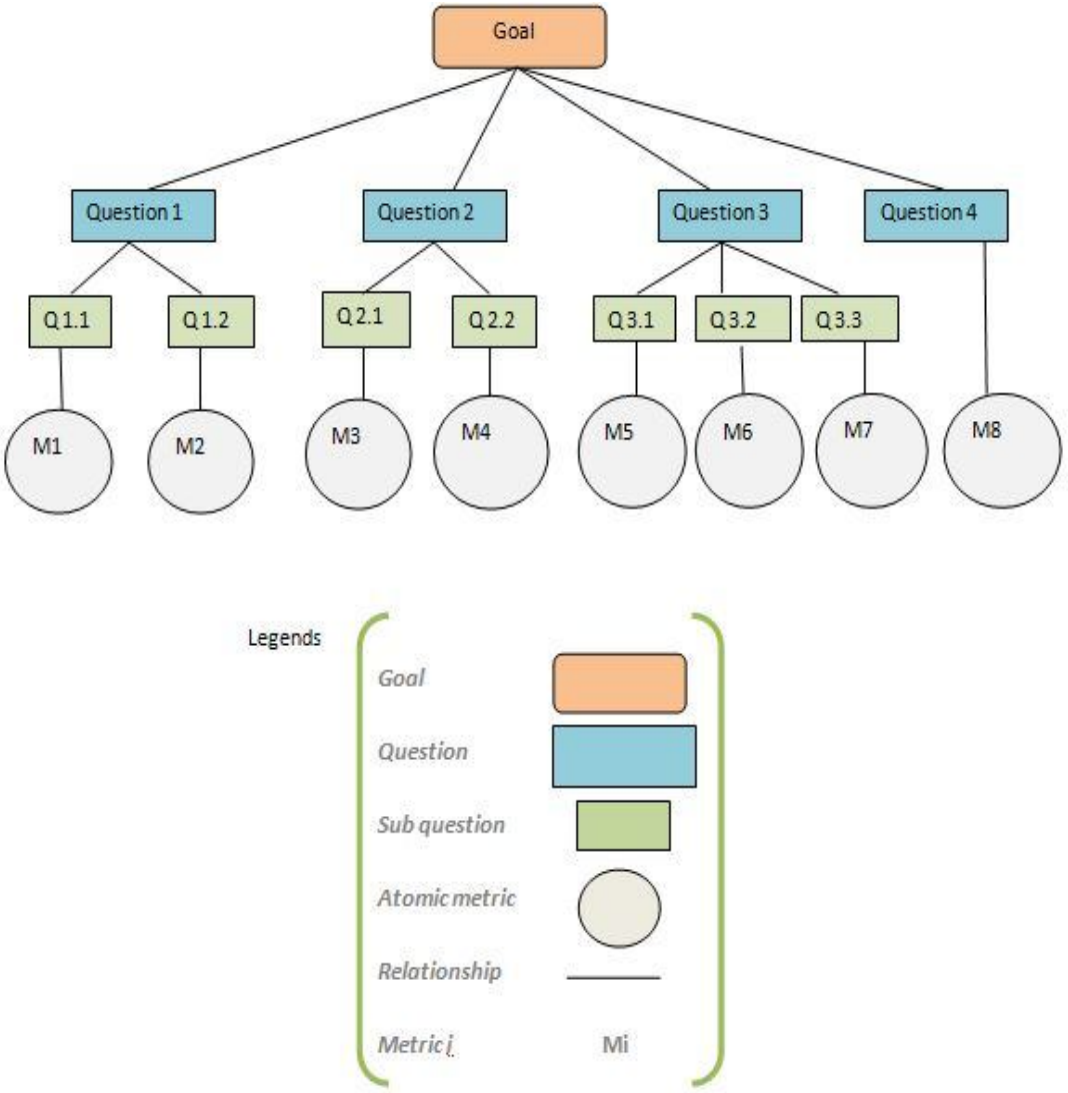
We have formulated the questions stated above by mapping different question formats mentioned by Yin in [60] and the possible substances of interest from these two dimensions. This will compliant our research questions with the goal of our study. Table 12 shows the possible substances of interest and their corresponding form of questions. We have also given the IDs of research questions and their associated metrics in Table 12 to demonstrate that the research questions do satisfy the goal.

It is important to mention that the metrics selected to satisfy the questions were limited to the scope of the study. For example, the impact of requirements missing information, on other project parameters such as costs (e.g., documentation costs, development costs, rework costs, etc.), product quality, product maintenance, RE-Success factors (e.g., the clarity of the business process in the architecture, the extent of user consensus on the recommended solution, the completeness of coverage of the cost/benefits analysis, etc.) [18], etc. has not been investigated in the current work and we intend to examine it in our future works.

Table 12: Possible substances of interest to satisfy the goal and their corresponding research questions, metrics

Parts of goal	Question format	Substance of interest	Research questions ID	Metrics ID
Missing information	*What extent*	Requirements (e.g., data, constraint, functionality etc), and requirements attributes (e.g., priority, dependency etc.)	Q1	M1, M2
	What extent	Types of changes (e.g., add, delete, modify)	Q2	M3, M4
	What	Characteristics of changed requirement (see)	Q3	M5, M6, M7
Impact of requirements and its attributes found missing	*What*	On software architecture in terms of effort	Q4	M8

Figure 3: Relationships between the goal, questions and metrics



3.2 Requirements document

The system was in the "banking" domain that was given for architecting. The application included three different modes of banking for the clients: ATM, internet banking and telephone banking services; client and financial database; various quality

drivers, such as security, availability, performance, usability, maintainability, and others. In total, there were some eighty nine high-level requirements to contend with, which is sizeable. The requirements followed the organisational structure as found in [54]. The requirements section was split into different sub-sections each detailing requirements for a given subsystem. Different properties of requirements are also defined like; different requirements' attributes information that the overall system should have. Prior to the start of the project, the architects were given sessions where the project was described, including the application domain and structure of the requirements, and any questions or concerns were addressed. Prior to conducting the study, these requirements were validated by two people for acceptability in general, and the semantic content of the document was not altered. The result of this process is that a few grammatical fixes were made, along with the clarification of certain requirements.

The validators had requirements, architecture, and software engineering experience ranging from 3 to 27 years. We did this in order to reduce *researcher bias* in the study. In a real world setting, requirements documents given for architecting or development of a system are not always *perfect*. We also did not want to "fix" the document to the point where it was considered *perfect.* By doing this, we intended to emulate this by delivering an acceptable document to the participants.

3.3 Architecting process

Given these requirements document, as mentioned in section 3.2, each of the five groups developed architecture from the same requirements using the ADD method [5]. The project was conducted at UWO. The key steps of the ADD method include: understanding the requirements, developing the quality scenarios; iteratively decomposing a selected module, choosing architectural drivers from the scenarios and functional requirements, choosing or creating an architectural pattern (by using appropriate tactics) that satisfies the architectural drivers, identifying child modules to

implement the tactics, defining interfaces, verifying and refining use cases and quality scenarios and making them constraints for the child modules.

Each team had to develop and document the system architecture. Besides, they had to fill out templates that captured data regarding requirements and requirements' attributes they found missing, characteristics of those missing information, impact in terms of effort spent on those missing information etc. In addition, each team had the freedom to seek help on any difficulties they faced during their project.

In the architecting projects the participants' architectures produced were *conceptual*. That is, there was not any implementation and so certain static properties (such as fitness between the architecture's structure and allocation of code components) and dynamic properties (such as delivery of performance, security, availability, etc.) couldn't be checked through actual implementation of the system. Due to that, there were no end-user consequences of the architectural decisions being made. The only possible consequences were academic-performance-related consequences.

3.4 Research Procedures

The type of study conducted was a *multiple-case* study design [15]. There were five parallel cases (i.e., one case per architecting team) in our study. Despite that, it was an *exploratory* study in that we had no initial hypothesis, and we did not know which phenomena were important (i.e., impact of missing information on architecture in terms of effort). This is because, to our knowledge, there wasn't much background literature related to the posed research questions. The exploratory nature of the case study allows for analysing the commonality and differences across cases that have similar traits [15]. It helps to do comparison between different groups' data for specific research questions.

3.4.1 Participants

We have used convenience sampling [33] to involve 20 students; undergraduate and graduate students in the study. In order to conduct the study involving students we received consent from the ethics board at The University of Western Ontario. The threat of using students as participants is discussed in the external threats to validity section (see section 3.5). The participants were randomly assigned to groups of four, making a total of five groups.

To ensure that the participants had sufficient knowledge to conduct the project, they were given theory knowledge in RE and assignments that allowed them to learn and familiarize themselves with RE practices such as elicitation, analysis, negotiation, validation, and prioritisation. We did not do analysis of participant backgrounds impact on our findings. Distributions of participants of different groups are given in Table 13.

Table 13: Distribution of participants' background knowledge and experience for different groups

Participant group	**Total persons in each group (4)**		**Number of participants taken SE courses before**	**Number of participants had professional SE experience from industry**
	Undergraduate students	**Graduate students**		
Group 1	**3**	**1**	**1**	**1**
Group 2	**3**	**1**	**1**	**2**
Group 3	**3**	**1**	**2**	**1**
Group 4	**2**	**2**	**2**	**2**
Group 5	**3**	**1**	**2**	**2**

3.4.2 Data collection

Each group was given two templates named *Requirements Missing Information Template (RMIT)* and *Updated Requirements Template (URT*) in addition to collecting data related to our research goal. The other two templates *Time Log Template (TLT)* and *Decisions, Issues, Rationale Template (DIRT)* were given to keep time and design decisions related information. In all the templates, possible terminologies were defined and the questions such as "Who should fill out the templates and when", were mentioned at the start of the templates itself. Descriptions of the templates are given in Table 14.

Table 14: Description of different data collection templates

Template Name	**User**	**Purpose and Summary of Instrument**
Requirement Missing Information Template (RMIT)	Participant groups	RMIT has questions about missing requirements information, their criticalities, architectural relevance etc.
Updated Requirements Template (URT)	Participant groups	URT has questions regarding different types of changes made in requirements information, characteristics of that information, effort in terms of rework on missing requirements information etc.
Time Log Template (TLT)	Participant groups	The participants filled the time spent on any project related activity in this form on an ongoing basis.
Decisions, Issues, Rationale Template (DIRT)	Participant groups	Each team had a team DIRT and each individual member of the team had their own DIRT. The DIRT was used so that

		participants could enter more qualitative data: all their design decisions, project issues and rationale relating to the project. They filled this d document on an ongoing basis during the project.

3.4.3 Data analysis

Our data collection templates include both nominal and ordinal scale data and they are mostly qualitative in nature. To calculate all atomic metrics, we have used percentage and frequency distribution [45]. Frequency distribution shows how frequently each value of a variable occurs in a set of scores and instead of showing actual number of occurrences of values within an interval, percentage distribution shows the occurrence in that interval as a percentage of total number of occurrences in the set [45] [55]. We have calculated metrics M1, M2, M3, M4, M5, M6, M7, and M8 (see Table 12) by using frequency and percentage distribution.

3.5 Threats to Validity

3.5.1 Internal Validity

Internal validity deals with whether we can infer that a relationship between two variables is causal, and not due to any confounding factors [33]. There is no cause and effect relation between variables that is applicable to our study. So we did not discuss this issue in internal validity. There are other numerous specific types of internal validity threats [33], we discuss here only the threats that applied to our study and the procedures we employed to contain the threat.

3.5.1.1 Differential selection:

It is possible when characteristics of the subjects by chance, differ between the two types of groups and possibly affect the quality of the data. In our study, such a characteristic is the participants' software engineering educational and industrial-experience backgrounds. Participants with differing SE background could possibly perform differently in the project. We have contained this threat by ensuring participants background and experience defined in section 3.4.1 (Table 13).

3.5.1.2 Differential mortality:

It occurs when a physical or mental change occurs to participants during study that is not "equal" between the two types of study groups. This threat existed in our study because of the duration of the participants' project, which lasted approximately two months. To contain this threat, the researchers assessed weekly submissions of work and collected data. At the conclusion of the study, all initial participants remained in the study and no effects of the differential mortality threat were observed.

3.5.1.3 Researcher bias:

It occurs when the researcher, knowingly or unknowingly, influences the outcome of the study. This threat exists in our study because of the subjective nature of the requirements definitions in the templates. To mitigate this threat, multiple researchers were used in the study processes. This is quite recognized technique for dealing with *researcher bias* [33].

3.5.2 Qualitative Validity

In qualitative studies, a validation technique, called *triangulation* [26], is used to ensure validity in the study. *Triangulation* is a method of establishing the accuracy of a

study's findings by comparing three or more types of independent points of view on a given aspect of the research process (methodology, data, etc.) [26]. There are different types of *triangulation* that can be used together to form a strong basis of validity. In this section, we will discuss how we used three different types of *triangulation* to ensure validity in our study. The *triangulations* used were: *data triangulation, methodological triangulation,* and *investigator triangulation*.

3.5.2.1 Data Triangulation

Data triangulation is the use of different sources of data/information on which the study results are based. If there is consistency in the data/information provided across the various data sources that are used, then this suggests that the data is valid. In our study, as mentioned in section 3.4.2 (Data Collection), our data-set came from various data collection templates that the participants had to complete. The proportions of the found requirements missing information were having some similarities in each of the sources.

3.5.2.2 Methodological Triangulation

Methodological triangulation is the use of different methodological techniques (that could be either quantitative or qualitative) in the study and, if the conclusions from each method are consistent, then validity is increased. In our study, we used various qualitative methods such as document analysis, as well as quantitative content analysis. The resultant data from these various methods, and its subsequent analysis, showed similar conclusions, that architects experienced problems with different missing requirements and requirements attributes when architecting the system (see chapter 4). This consistency establishes methodological validity in our study.

3.5.2.3 Investigator Triangulation

Investigator triangulation uses several investigators/researchers in the conduct of the study and all its processes. In our study, at every stage in the process (e.g., data collection, data analysis, research question validation, etc.), multiple researchers were involved to actually perform the processes as well as validate them. The findings observed to ensure that the conclusions are similar and therefore we conclude validity was reached.

3.5.2.4 Ecological Triangulation

Another type of triangulation that exists, but which we could not attain, is *Ecological Triangulation*. This is when the study is conducted at many different settings and places, and then the findings from each of these settings/places are compared to see if they are similar. This type of triangulation can be attained for this study through replication of this study in other contexts (e.g., in industry). Without first replicating this study, it is difficult to generalise the results to other contexts. However, this research provides a necessary groundwork for further studies of this kind.

3.5.3 External Validity

External validity is the degree to which any findings from the study can be *"generalised to and across populations of persons, settings, and time"* [15]. Using students as participants in our study is a threat that is directly imposed on the *generalisability* of the findings to industrial contexts.

3.5.3.1 Population validity

Using students as participants in our study is a threat that is directly imposed on the *generalisability* of the findings to industrial contexts. This is a common risk in ethnography based studies. Recent research in software engineering [30][50][49] has

shown that senior-level students perform similarly to “novice” software engineers with one-two years industry experience. Also, in [7], the use of students is promoted when conducting an investigation that has not been studied much before, such as in our case. Studies with students can provide early indications of trends, and preliminary evidence prior to committing to conducting studies in industry.

We did not include *temporal validity* threat here because there is little reason to believe that the results of this study could not be generalised over time given the current set of requirements methods, tools, processes, etc. that requirements engineers use.

3.5.4 Construct Validity

For construct validity, correct operational measures need to be established based on theoretical constructs for the concept being studied [60].We have constructs for different research questions defined in section 3.2 .Two types of construct validity which are applicable to our research are given as follows:

3.5.4.1 Content validity:

It is concerned with whether the research instrument (in our case the data template questionnaires) properly represents the specific intended domain of content [47].All the constructs and metrics related to research questions (i.e., M1, M2, M3, M4, M5, M6, M7, M8) are derived from information from books [36][5]. Requirements information is gathered from [25]. So, all the contents of the parameters are rooted in literature.

3.5.4.2 Face validity:

It is concerned with whether any contents (in our case the questionnaires) used for conducting the study are appropriately translated from the construct [47]. All data are captured based on constructs defined with each research questions (see section 3.2) and all the constructs are represented in the questionnaires as well. So there is nothing more and nothing less. This is validated in our study by involving two researchers for reviewing the questionnaire, in terms of both contents and its structure.

3.5.5 Conclusion Validity

Conclusion validity is about whether the conclusions we make, based on the findings, are reasonable [47]. To ensure conclusion validity is contained, two accepted principles that were applied to the study were ensuring reliability of data measurements and adequate implementation of study processes. To ensure the reliability of data measurements, we utilized data-collection instruments (see Table 14) that were validated by multiple experts. To ensure adequate implementation of study processes, sessions were held with the participants to explain the templates and study processes to them. In chapter 4, we have demonstrated that all our conclusions are rooted in the results and our findings are tested using statistical analysis, thereby maintaining the conclusion validity. Thus we can claim our conclusions are traceable through data analysis all the way to the research questions.

Chapter 4: Results and Interpretations

4.1 Missing requirements and requirements' attributes found during SA

We have considered both requirements and requirements attributes' information which are found missing while architecting. Requirements information are discussed in section 3.1 (see Table 1) and requirements' attributes are also defined in section 3.1 (see Table 3). We have discussed findings about each one of them as follows:

4.1.1 Missing requirements found while architecting

Table 15 shows the numbers of requirements information that are found missing while architecting the system for different groups. The first column includes different types of requirements' information defined in the study [25] (see section 3.1) (Table 1). For individual groups, frequency counts and percentage values, both are given, that are found missing while architecting.

With reference to Table 15, the top three types of missing requirements' information that are found missing in requirements while architecting are**:**

- Constraint information
- System functionality information
- System interaction (users/systems) information

Table 15: Number of requirements found missing, while architecting the system

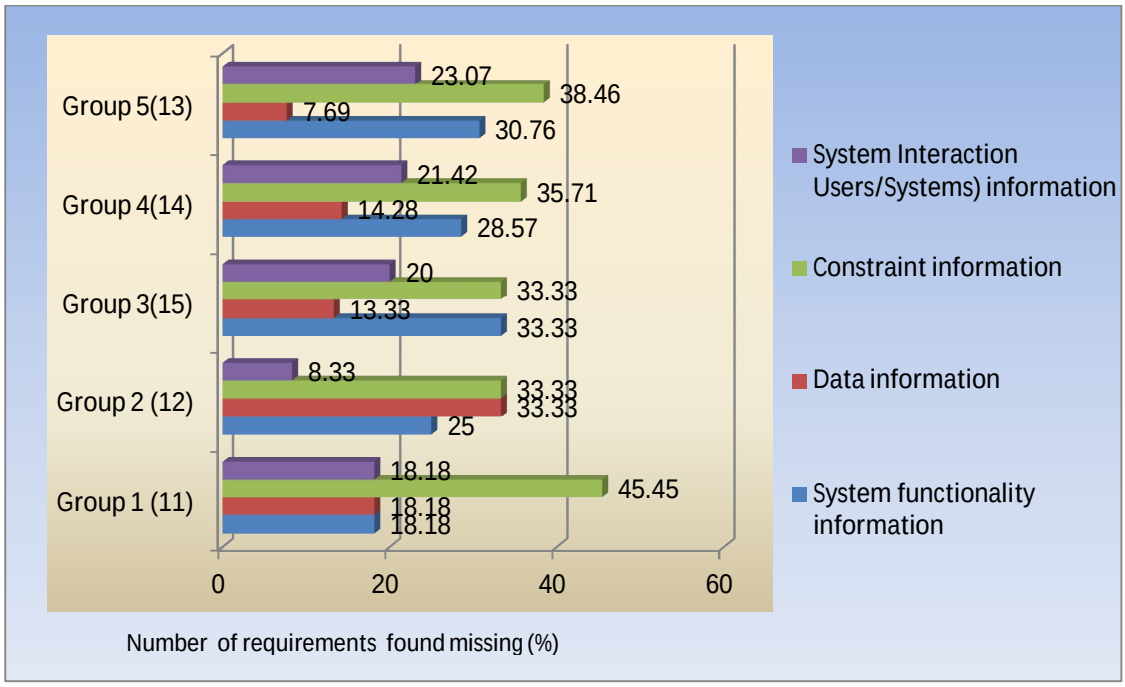
Types of requirements information	**Group 1**	**Group 2**	**Group 3**	**Group 4**	**Group 5**
	n (%)	**n (%)**	**n (%)**	**n (%)**	**n (%)**
Total	11	12	15	14	13
System functionality information	2(18.18)	3(25)	5(33.33)	4(28.57)	4(30.76)
Data information	2(18.18)	4(33.33)	2(13.33)	2(14.28)	1(7.69)
Constraint information	5(45.45)	4(33.33)	5(33.33)	5(35.71)	5(38.46)
System Interaction Users/Systems) information	2(18.18)	1(8.33)	3(20)	3(21.42)	3(23.07)

Number of requirements found missing by different groups:

From Figure 3, we can see that, *constraint information* was mostly found missing in requirements for all the groups. For group 1, they have found a total of 11 missing requirements and among them 45.45% are related to *constraints* of the system. For group 3, group 4 and group 5, all of them have found *constraint information* missing in requirements at higher percentages; that are, 33.33%, 35.71%, and 38.46% respectively. Other than *constraint* information, *system functionality information is* also found missing in requirements for most of the groups. Out of 15 missing requirements, group 3 has found 33.33% of missing requirements due to missing *system functionality information*. Group 4 has found a total of 14 missing requirements and among them 4 are related to *system functionality information*; that is, 28.57% of the total missing requirements. On the other hand, group 5 has found 4 missing requirements out of a total of 13 missing requirements; that is, 30.76% of the total missing requirements. System *data information*

was found missing less frequently among all the groups except group 2, that is 33.33% of total missing requirements.

Figure 4[2]: Number of requirements missing, while architecting the system



4.1.2 Requirements' attributes found missing, while architecting

In Table 16 numbers of different requirements attributes are given, which are found missing while architecting. The first column lists different requirements attributes as defined in section 3.1 (see Table 3). For individual groups we have given both the frequency counts and the percentage values.

Table 16 indicates that, the top three requirements attributes that are found missing while architecting are**:**

[2] For each of the groups, total number of missing requirements found by them is given within braces. For example, group 1(11) means they have found in total 12 missing requirements

- Priority
- Rationale
- Dependency

Table 16: Number of requirements attributes, found missing

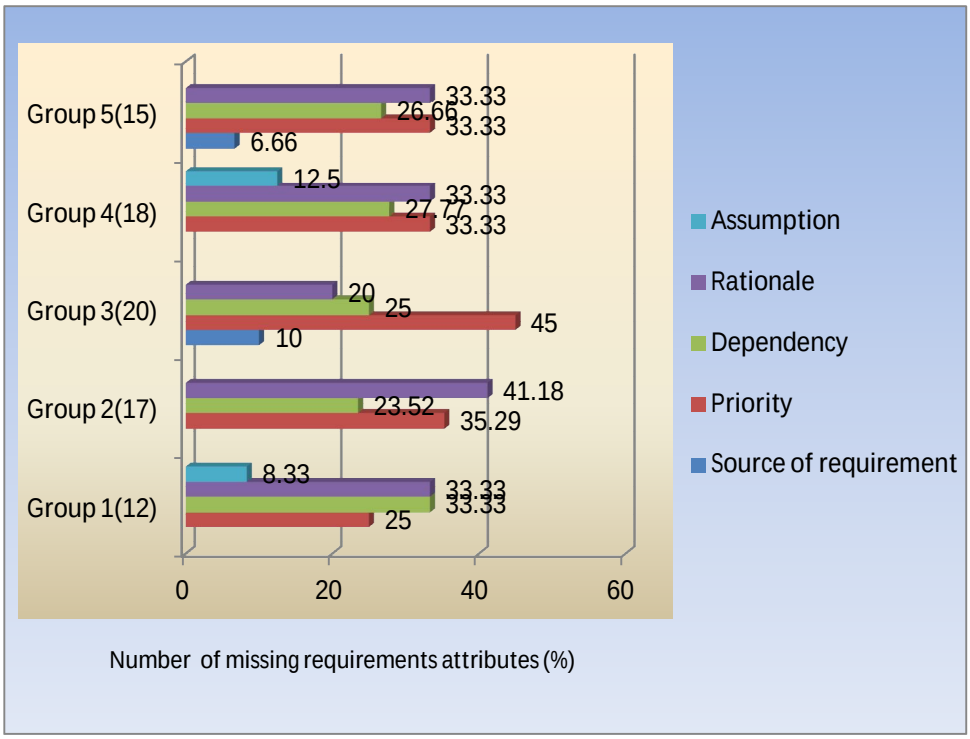
Requirements attributes	Group 1	Group 2	Group 3	Group 4	Group 5
	n (%)	**n (%)**	**n (%)**	**n (%)**	**n (%)**
Total	12	17	20	18	15
Source of requirement			2(10)		1(6.66)
Priority	3(25)	6(35.29)	9(45)	6(33.33)	5(33.33)
Dependency	4(33.33)	4(23.52)	5(25)	5(27.77)	4(26.66)
Rationale	4(33.33)	7(41.18)	4(20)	6(33.33)	5(33.33)
Assumption	1(8.33)			1(12.5)	

Number of requirements' attributes found missing by different groups:

It is found that (Figure 5), all the groups more or less found requirements' attributes missing that were needed while architecting the system. Among them, *priority* was missing at higher percentages for group 3 (45%), group 2 (35.29%), group 4 (33.33%), and group 5 (33.33%). Group 4 has found 6 missing attributes related to priority, out of 18 total missing requirements attributes and group 5 has found 5 missing requirements attributes related to *priority* out of 15 missing requirements attributes. If we interpret the findings for group 4 and group 5, both of them found one *priority* related information missing in every three missing requirements attributes. For group 2, out of total 17 missing requirements attributes found 7 are related to *rational*; that is, 41.18% of the total. *Rationale* is also found missing at a higher percentage for group 4 (33.33%) and group 5 (33.33%), out of total missing requirements attributes they have found. Requirements *dependency* was also found missing for all the groups to some extent.

Among them, group 1 found it at a higher percentage that is 33.33% of the total; that is, 4 out of 12 missing requirements attributes. Requirements *assumption* and *source of requirements* are found missing at a lower percentages for some of the groups. Group 3 and group 5 have found *source of requirements* missing at a percentage of 10% and 6.66%, out of their total number of missing requirements attributes respectively.

Figure 5[3]: Number of requirements attributes' missing, while architecting



4.2 Number of changes made in requirements and requirements' attributes during SA

We have considered three types of changes (i.e., add, delete, modify) in order to categorize different changes made to requirements and requirements' attributes. Both are described as follows:

[3] For each of the groups, total number of missing requirements attributes found by them is given within braces. For example, *group 1(12)* means they have found in total of 12 missing requirements attributes.

4.2.1 Number of changes made in requirements

In Table 17 numbers of changes that architects did on requirements are given. For individual group, frequency counts and percentage values, both are given. It is found that other than *adding* or *deleting* requirements, all the groups *modified* requirements the most. Number of changes were higher for group 3 (15) and group 4 (14), out of total number of requirements.

Table 17: Number of changes in requirements

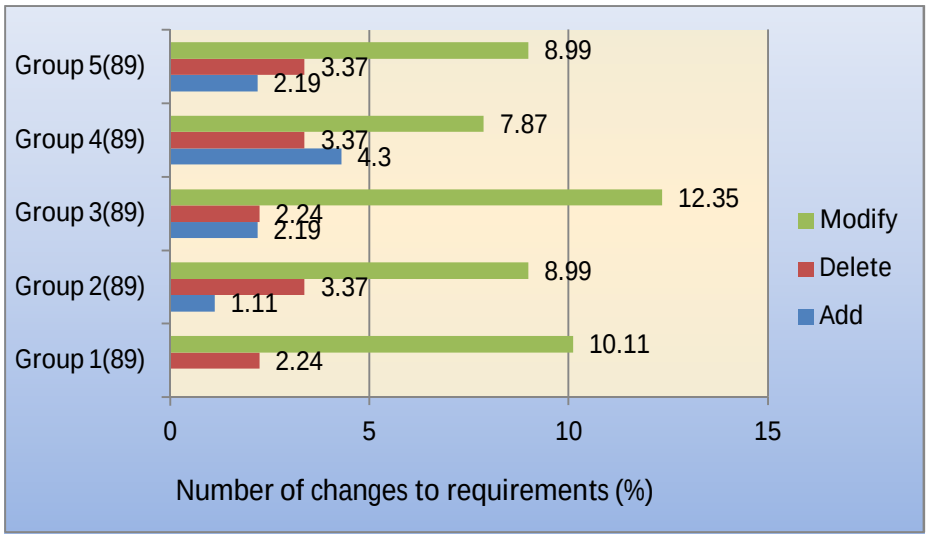
Types of changes	**Group 1**	**Group 2**	**Group 3**	**Group 4**	**Group 5**
	n (%)	**n (%)**	**n (%)**	**n (%)**	**n (%)**
Add		1(1.11)	2(2.19)	4(4.30)	2(2.19)
Delete	2(2.24)	3(3.37)	2(2.24)	3(3.37)	3(3.37)
Modify	9(10.11)	8(8.99)	11(12.35)	7(7.87)	8(8.99)
Total number of requirements	89	89	89	89	89
Total number of changes in requirements	11	12	15	14	13

Number of changes made to requirements by different groups:

Figure 6 depicts different types of changes made in requirements by architects' of different groups. Among different groups, *modification*s of requirements are found the most. Group 3 had the highest number of modifications to requirements that is 12.35% of total requirements given to them. In every eight requirements, given to them, they have modified one requirement. After group 3, group 1 had a higher number of *modifications* of 10.11% out of total requirements. Among all, group 4 did least number of *modifications* that is 7.87% of total requirements. In the case of *deleting* requirements, group 2, group 4, and group 5 had the same numbers of *deletions* that is 3.37% of the total requirements. Among all the groups, *addition*s of new requirements are less

frequent. Group 4 had a higher numbers of *additions* than other groups that is 4.30% of total requirements. It is found only in case of group 1 that they did not add any new requirements.

Figure 6[4]: Number of changes made to requirements, while architecting



4.2.2 Number of changes made in requirements' attributes:

In Table 18 the numbers of different types of changes that architects' made on requirements attributes are given. For individual group, frequency counts and percentage values, both are given. It is seen that other than adding or deleting requirements, all the groups *modified* requirements attributes the most. The number of changes are higher for group 3 (13) and group 2 (12), out of a total number of requirements given to them.

[4] Percentages are calculated in compare to total requirements given to each group that is 89. In case any group added any new requirement, percentages are calculated based on the formula ((n/(n+ total number of requirements))*100 %), where n is the number of new requirements.

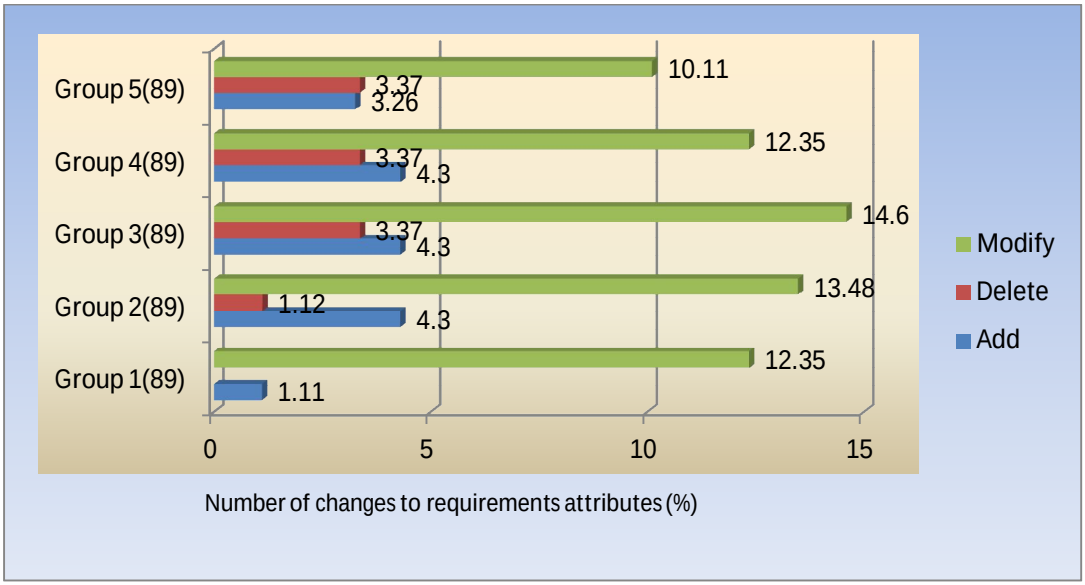
Table 18: Number of changes in requirements attributes

Types of changes	**Group 1**	**Group 2**	**Group 3**	**Group 4**	**Group 5**
	n (%)	**n (%)**	**n (%)**	**n (%)**	**n (%)**
Add	1(1.11)	4(4.30)	4(4.30)	4(4.30)	3(3.26)
Delete		1(1.12)	3(3.37)	3(3.37)	3(3.37)
Modify	11(12.35)	12(13.48)	13(14.60)	11(12.35)	9(10.11)
Total number of requirements	89	89	89	89	89
Total number of changes in requirements attributes	12	17	20	18	15

Number of changes made to requirements' attributes by different groups:

Figure 7 shows the different types of changes made by architects of different groups on requirements attributes. Among the different groups, *modifications* of requirements attributes are found the most. Group 3 made the highest number of *modifications* to requirements attributes; that is, 14.60% of total requirements given to them. In every seven requirements they have made one modification to requirement attribute. After group 3, group 2 made a higher number of *modifications;* that is, 13.48% out of total requirements. Among all, group 5 did least number of *modifications* that is 10.11% of the total requirements. In case of *deleting* requirements attributes, group 2, group 4, and group 5 had the same numbers of *deletions*; that is, 3.37% of the total requirements. As an exception, group 1 did not have any *deletion* on requirements attributes. Among all the groups, *additions* of new requirements attributes have been made to some extent. Group 3, group 4, and group 5 made the same numbers of additions; that is, 4.30% of the total requirements. Group 1 did only 1.11% addition of requirement attributes out of total requirements.

Figure 7: Number of changes to requirements attributes, while architecting



4.3 Characteristics of the changed requirements

In Table 19, numbers of positive responses to different characteristics questions of the changed requirements (see section 3.1) (Table 6) are given, from the architects' points of view. The first column includes the characteristics of the changed requirements that we have considered for our research are given. For individual group both their frequency counts and percentage values are given in Table 19.

Table 19: Frequency of different characteristics of the changed requirements

Requirement characteristic	**Group 1**	**Group 2**	**Group 3**	**Group 4**	**Group 5**
	n(%)	**n(%)**	**n(%)**	**n(%)**	**n(%)**
Architecturally significant requirement	9(60)	8(44.44)	8(38.09)	7(35)	8(44.44)
Crosscutting requirement	8(53.33)	10(55.55)	12(57.14)	10(50)	9(50)
Quality requirement	9(60)	10(55.55)	11(52.38)	9(45)	8(44.44)

Total number of changed requirements[5]	15	18	21	20	18

Number of architecturally significant requirements found by different groups:

With reference to Figure 8 we found that, all the groups, more or less, found changed requirements as architecturally significant. Among them group 1 defined 9 out of 15 changed requirements as architecturally significant; that is, 60% of total changed requirements. That means in every one or two changed requirements, one was architecturally significant from their point of view. Group 4 found the least number of changed requirements as architecturally significant; that is, 21.8% of the total changed requirements.

As an example, there was a requirement that said,

"The system should respond faster to any request"

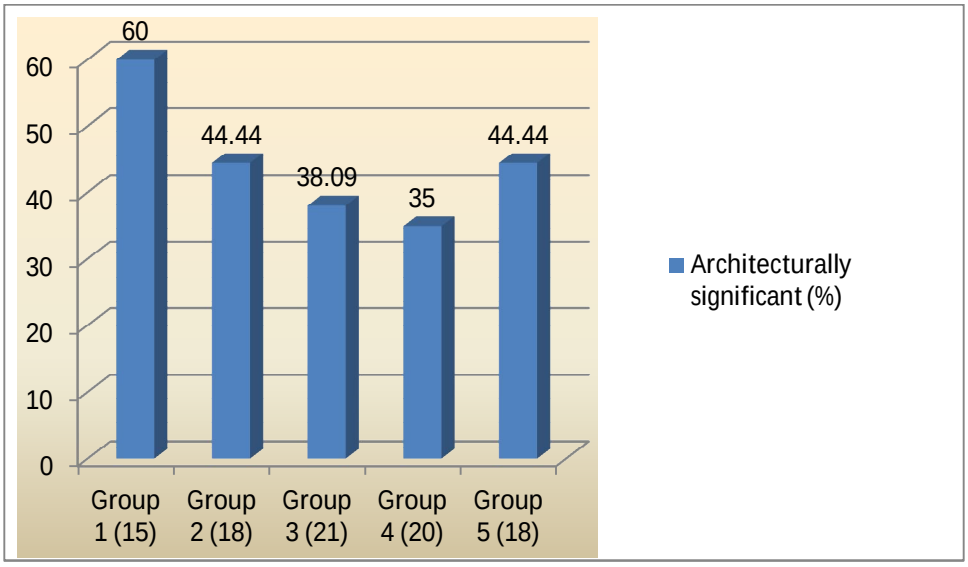
Architects could not deal with the word "faster" so they thought of a standard response time for any request and changed the requirement to

"The system should respond within 5-10 seconds for any request"

If we look at the instance level of this requirement, it deals with both performance and security aspect of the system. So from an architectural point of view, it will touch different components of the architecture and based on that different hardware resources will be allocated to it. So from architects' points of view they defined this changed requirement as an architecturally significant requirement.

[5] For individual group, if we sum up the frequencies of different characteristics of changed requirements, it will differ with total number of changed requirements because one requirement may satisfy one or more characteristics at a time. For example, a requirement can be crosscutting and quality requirement at the same time.

Figure 8: Number of architecturally significant requirements



Number of crosscutting requirements found by different groups:

Figure 9 depicts that, all the groups to some extent found changed requirements are crosscutting requirements. Among them group 3 has found 12 changed requirements as crosscutting requirements out of 21 changed requirements; that is, 57.14% of total changed requirement. If we interpret this in another way, in every three changed requirements they have found one as a crosscutting requirement. After them, group 2 and group 3 have found 55.55% and 53.33% of their individual total changed requirements as crosscutting requirements.

As an example, there was a requirement that said,

"A customer must be able to access their account by using 5 digits PIN"

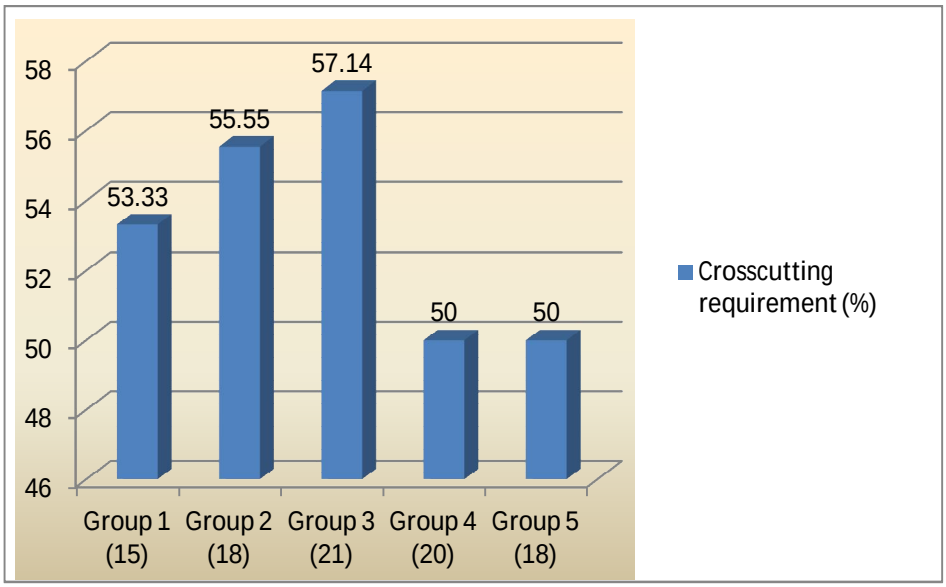
The priority was given to this requirement as *"medium"*. But architects' found this feature very important from architecture's perspective as other requirements are dependent on it like:

"A customer must be able to pay a bill using the ATM"

"A customer must be able to withdraw funds from his/her account(s) using the ATM"

So it is found that, architects changed the priority of the requirement to "very high" as from the architects' points of view this requirement is crosscutting other requirements and if any problem occurs with this requirement other security features of the system could be affected as well.

Figure 9: Number of crosscutting requirements



Number of quality requirements found by different groups:

From figure 9.we found that, all the groups found to some extent changed requirements as quality requirements. Among them group 1 has found 60% of total changed requirements as quality requirements. They have found 9 changed requirements as quality requirements out of 15 changed requirements; that is, in every two to three changed requirements one is a quality requirement. Group 2 and group 3 have found

55.55% and 52.38% of changed requirements as quality requirements out of their individual total of changed requirements.

As an example, there is a requirement that says,

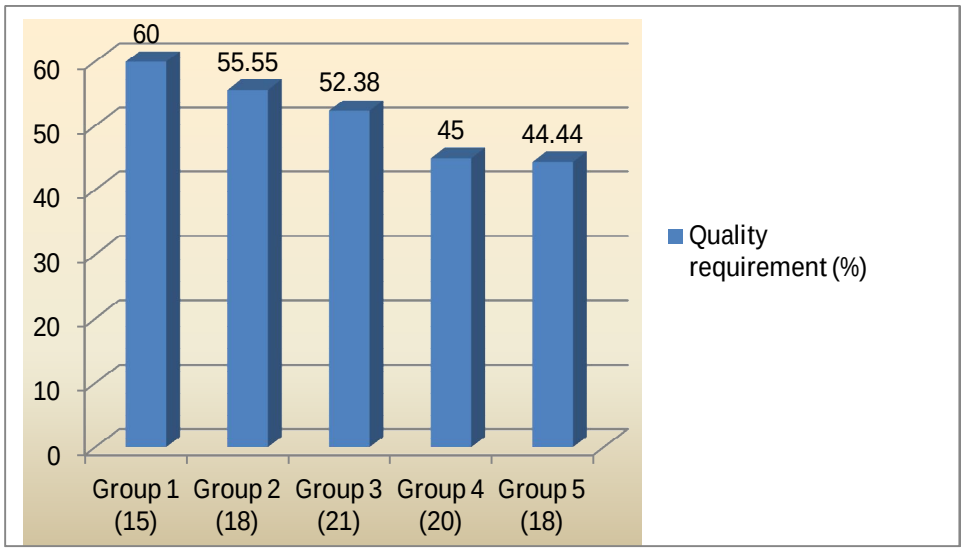
"Backups should be taken regularly"

Architects' have modified this requirement as,

"Backups should be taken in every 12 hours"

This is an important security feature and it also deals with reliability of the system as well as different banking domains (i.e., ATM, internet banking etc.) are interrelated with each other. That will help architects' to allocate resources to it in a more efficient way and it will increase both the security and reliability aspect of the system. Both of these qualities *security* and *reliability* are considered as quality attributes of a system. So architects' defined this changed requirement as a quality requirement.

Figure 10: Number of quality requirements



4.4 Impact on rework of requirements missing information, in terms of effort while architecting

Effort spent on different missing requirements and requirements' attributes

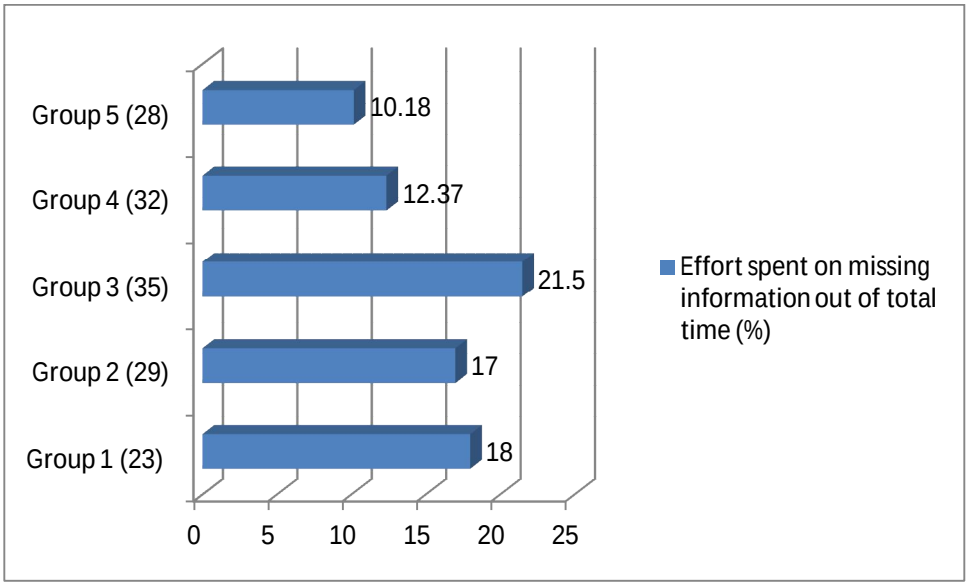
Table 20 depicts effort spent on recreating different requirements and requirements attributes' missing information. We have calculated the overall time from the time log template (TLT) (see section 3.4.2) (Table 14) and compared the effort spent on recreating the missing information with that. It is found from Table 7 that all groups did spend certain portion of the total time only in recreating missing requirements and requirements attributes' information, while architecting. Among them group 3 spent the most that is 21.5% of the total time they spent on the project. They took this amount of time for making around 35 different changes that includes both requirements and requirements attributes' missing information. Whereas, group 4 spent 12.37% of time for making a total of 32 changes to requirements and requirements attributes missing information. Group 1 has spent 18% and group 2 spent 17% of the total time in recreating missing information while architecting. Group 5 spent the least amount of time among all the groups; that is, 10.18% of total time they spent on the project. It seems from the findings (see Table 20) that identified missing requirements information has impact on different groups in terms of effort spent on recreating that information

Table 20: Effort spent (person-hours) on recreating missing information

Participants	Total number of changes made to requirements and requirements attributes[6]	Overall Time spent on architecting the system (hours)	Time spent on rework on documents (hours)	Proportion of entire project (%)
Group 1	23	24	4.33	18
Group 2	29	31.8	5.46	17
Group 3	35	29.66	6.4	21.5
Group 4	32	35	4.33	12.37
Group 5	28	32	3.26	10.18

[6] To calculate total number of changes to requirements, we have counted requirements change and requirements attributes' changes separately so that will differ with total number of changed requirements.

Figure 11[7]: Effort spent on missing information while architecting



[7] For individual group, total number of changes is given within braces.

Chapter 5: Implications

This chapter discusses implications of our results. Sections 5.1, 5.2 and 5.3 discuss the implications on practice, tools and empirical research**.**

5.1 Implications on practice

Our investigation finds different missing requirements and their attributes, and to what extent they are found missing during SA. Even though the requirements are elicited by following standard RE process (i.e., elicitation, analysis, validation etc.), there is missing information found by architects that is important from architects' points of view. Besides, in practice, requirements engineering methods are inconsistently followed [21] and sometimes they are based on a single technique (i.e., use case modeling) and are used for all types of requirements information [21] that ends up with poor products. The knowledge about what requirements information and requirements attributes are important (see sections 4.1.1 and 4.1.2) for the architecting purpose will help practitioners to follow RE processes properly so that minimal information goes missing.

Our findings also show the effect of this missing information on SA in terms of effort (see section 4.4).In practice, missing requirements are often missed until the system is integrated or in the worst case after deployment [21]. So our findings will help practitioners at the early phase of development by providing an idea about what kinds of information about requirements are important and what affect it could have on architecture,. Eventually it will help in the "handover" process, from requirements engineers to the architects in a more structured way and emphasis should be placed on ensuring that the architects not only understand the requirements as documented, but have comprehended them in terms of architecting different types of projects. For example, during the development of a safety critical system, or a system which includes regulatory requirements from the customer, the requirements should be documented

carefully, keeping in mind the different requirements and requirements attributes' missing information from our findings that could help the architects of the system.

5.2 Implication on Tools

There are already some research tools that are available (e.g., [23] and EGRET [51]) and tools used in industry (e.g., Rational Team Concert [46]), which provide traceability between software artefacts and communication artefacts (e.g., meeting videos, email, chat, etc.). In practice, there is inadequate tool support for engineering requirements[21] and that makes the task of maintaining requirements and other artifacts *extremely* labor-intensive [21][30]. In reality, simple and powerful documentation tools and formats are needed [40]. Our findings could be useful in developing a tool or prototype of a tool that will eventually try to mitigate certain problems from our findings with requirements and requirements attributes or improving features of different tools.

5.3 Implication on empirical research

Based on the findings of this study, we propose the following hypotheses which might be tested through further empirical investigation.

H1: *Requirements attributes' information is missing more than requirements themselves, in requirements documents.*

Rationale: With reference to chapter 2 (section 2.6), we could see requirements information goes missing in RE documents and it has impact on SA. Section 4.1.1 (Tables 15) shows the extent to which requirements information found missing and section 4.1.2 (Table 16) shows the extent to which requirements attributes are found missing. Even though requirements information is found missing to certain numbers for all the groups, requirements attributes' information is found missing for all the groups at

higher numbers. For example, group 2 found a total of 12 requirements information missing but they found a total of 17 missing requirements attributes' information. It is similar for all the groups more or less. These findings motivate us to state the above hypothesis.

H2. *Software architects modify requirements and requirements attributes more than adding or deleting requirements.*

Rationale: Section 4.2.1(Table 17) shows the extent of different changes made to requirements and section 4.2.2 (Table 18) shows the extent of different changes made to requirements attributes' information. It is found for all the groups that architects did modify the requirements and requirements attributes' information at higher numbers. For all the groups, they less frequently added or deleted requirements or requirements attributes' information. For example, group 3 modified 11 requirements out of 15 changed requirements and they also modified 13 requirements attributes out of 20 changed requirements attributes. These findings motivate us to state the above hypothesis.

H3. *Missing requirements and requirements attributes have impact on SA, in terms of effort on recreating missing information.*

Rationale: Section 4.4.1 (Table 20) shows the effort spent by different groups on recreating missing information while architecting out of total time they spent on the project. For all the groups they spent a certain amount of time in just recreating missing information out of total time they spent on the project. Even though for all the groups we could not prove it by any analysis that the time they spent was a significant amount of

time, but these values certainly cannot be ignored as well. These findings motivate us to state the above hypothesis.

To test these hypotheses, further empirical investigations would need to be conducted (i.e., case study, controlled study, survey, replicated study etc.). Also, our questionnaire and data analysis method can be considered as a primary template for investigating the impact of missing information on other areas of software engineering (i.e., coding, testing, maintenance, etc.)

Chapter 6: Limitations, Conclusions, Ongoing and Future Work

This chapter discusses the limitations of the study, and conclusions in Sections 6.1 and 6.2 respectively. Finally section 6.3 describes the ongoing and future work of our thesis.

6.1: Limitations

To best of our knowledge, the following is a list of the limitations of the study:

- As a research strategy, the case study has some inherent limitations. For example, data collected through the study reflects the participants' perception of the situation rather than the actual situation in practice. Because we used case study as our research strategy our study has the limitations of case study research.

- While measuring the impact of missing requirements and requirements attributes on SA, we only focused on impact on SA, in terms of effort and excluded other aspects such as cost, quality etc.

- Our research was only specified to the *banking* domain, so we could not generalize our finding for other domains.

- We did not classify our findings based on different quality attributes (i.e., performance, security etc.) rather we have considered the overall aspect by using the term *quality requirement*.

- We did not do the qualitative analysis of the data at the architectural level for different groups.

- We could not obtain the perception of the participants and other practitioners towards the findings of this study.

- We could not make a comparison between the values of our metrics in the back drop of contextual information such as RE process models followed, participants' background or experience etc.

All these limitations are mainly due to resource and time constraints. We intend to overcome these limitations in our future work. We also encourage other researchers to conduct confirmatory and complementary studies in other domains and contexts to help and build grounded theory on the characteristics of missing requirements and impact of this missing requirements and requirements attributes on SA.

6.2 Conclusions

Even though research evidences about missing information between the phases of RE and SA can be found in the literature (i.e., [24][25]), no scientific studies have been found on identifying the extent to which requirements and requirements attributes information is found missing during SA, characteristics and impact of those missing information in SA in terms of effort . In this thesis, we have described a case study on a classroom project, involving five groups in total. Our study investigated the extent of missing requirements and requirements attributes' information that is found while architecting; characteristics of changed requirements and impact of missing information on SA, in terms of effort.

We found *constraint information* and *system functionality information* was missing at higher numbers for different groups (see section 4.1.1) in requirements. Section 4.1.2 depicts different missing requirements' attributes and *priority, dependency*

and *rationale* have higher percentages among them. Section 4.2 (Tables 15 and 16) describes the extent of different changes made to requirements and requirements attributes. Among them *modification* has higher percentages for almost all the groups. Section 4.3 describes the characteristics (i.e., architecturally significant requirement, crosscutting requirement, and quality requirement) of the changed requirements for different groups. Section 4.4 shows the evidence of impact of missing information on SA, in terms of effort. Among all the groups, group 3 spent more time on missing information; that is, 21.5% of total time that was spent on the project.

Our results have implications in the industry as it could help the practitioners to understand the importance of the RE process. It also could help them to keep in mind different kinds of requirements information that should be taken care of during the RE process (see section 5.1). But we advise caution when making a business decisions based on the results of this fundamental study alone.

Our results also have implications for research as a numbers of new hypotheses emerge from it (see section 5.3). We encourage other researchers to conduct confirmatory and complementary studies in other domains and contexts to help build a grounded theory on different characteristics of requirement and impact of missing requirements information on SA, in terms of effort.

6.3 Ongoing and Future works

In this case study our findings are limited to missing requirements and requirements attributes and these results data are mostly quantitative in nature. We have shown here *to what extent* this information was missing but did not explain the details of the findings at the architectural level for different groups. We have presented findings about different characteristics of the changed requirements but could not analyse our data at the architectural level. That would possibly give a better insight about the rationales behind different changes and how these changes actually affecting the architecture.

Section 4.4.2 (Tables 21 and 22) shows that there is significance difference between groups in terms of effort spent on rework. This motivates us to investigate further on our data as well. These things will be done as an ongoing basis. From the data we have collected, the following emerging questions we will analyze further.

Q1. *How are different SA activities affected by missing requirements and requirements attributes?*

Q2. *How are different missing requirements and requirements attributes relevant to SA attributes, from architecture's point of view?*

Q3. *What is the impact of requirements and requirements attributes' missing information on SA, in terms of architectural quality?*

In addition, we will further explore this impact in other development phases to see, other than SA, to see how other development phases (i.e., coding, testing, product management etc.) are affected by missing requirements and requirements attributes' related information in terms of quality.

We do have a proof-of-concept prototype tool that was initially developed by one other student two years back. It provides traceability between meeting videos and software artefacts (i.e., requirements and architectural artefacts) based on the developer's (whoever was using the tool during meeting) action in the tool. The tool also provides the facility to develop traceability between different software artefacts through tagging (e.g., image tagging). Further enhancement of the tool will be done based on the findings of the analysis of project data mentioned above. The comparison of return on investment in terms of cost, quality and effort of using the tool for projects, through further empirical investigation is also part of future research plans.

References:

[1] Albayrak, O., Kurtoglu, H., & Biaki, M. (2009). *Incomplete Software Requirements and Assumptions Made by Software Engineers***.** 16th Asia-Pacific Software Engineering Conference, IEEE Software, pp. (333-339).

[2] Ambler, S. W. (n.d.). *Introduction to User Stories*. Retrieved November 2009, from Agile Modeling: http://www.agilemodeling.com/artifacts/userStory.htm.

[3] Bachmann, F., Bass, L., & Klein, M. (2003*). Deriving Architectural Tactics: A Step Toward Methodical Architectural Design*. Software Engineering Institute, pp. (35-45). CMU.

[4] Basili, V. R., Caldiera, G., & Rombach, H. (1994). *The Goal Question Metric Approach.* In J. J. Marciniak (Ed.), Encyclopedia of Software Engineering (2nd ed), Vol. 2, pp. (528-532). John Wiley & Sons, Inc.

[5] Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice (2nd ed.)*. Addison Wesley.

[6] Beecham, S., Hall, T., & Rainer, A. (2003). *Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis*. Empirical Software Engineering, *8* (1), pp. (7-42).

[7] Berander, P. (2004). *Using Students as Subjects in Requirements Prioritization.* Proceedings of the 7th International Conference on Empirical Assessment & Evaluation in Software Engineering, pp. (95-102).

[8] Berenbach, B. (2006). *Impact of Organizational Structure on Distributed Requirements Engineering Processes: Lessons Learned.* International workshop on Global software development for the practitioner, 28th International Conference on Software Engineering (ICSE 06), pp. (15-19). Shanghai, China: ACM.

[9] Bjarnason, E., Wnuk, K., Regnell, B. (2011). *Requirements are Slipping Through the Gaps – A Case Study on Cause & Effects of Communication Gaps in Large-Scale Software Development.* 19th IEEE International Requirements Engineering Conference, pp. (37–46).

[10] Boehm, B., & Bose, P. (1994). *A collaborative spiral software process model based on Theory W*. Third International Conference on the Software Process, pp. (59-68).

[11] Burge, J.E., Carroll, J.M., McCall, R., Mistrík, I. (2008). *Rationale and Requirements Engineering (3rd ed.)*, pp. (139-153). Springer.

[12] Cao, L., & Ramesh, B. (2008). *Agile Requirements Engineering Practices: An Empirical Study*. IEEE Software, 25 (1), pp. (60-67).

[13] Chen, L., Ali Babar, M., & Nuseibeh, B. (2013). *Characterizing Architecturally Significant Requirements.* IEEE Software, 30(2), pp. (38–45).

[14] Cleland-Huang, J., Czauderna, A., & Keenan, Ed. (2013). *A persona-based approach for exploring architecturally significant requirements in agile projects*.19th international conference on Requirements Engineering: Foundation for Software Quality, REFSQ ’13, pp. (18-33). Berlin, Heidelberg.

[15] Creswell, J. W. (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches.* Thousand Oaks, CA: Sage Publications.

[16] Damian, D., Izquierdo, L., Singer, J., & Kwan, I. (2007). *Awareness in the Wild: Why Communication Breakdowns Occur*. International Conference on Global Software Engineering pp. (81-90). Munich: IEEE Computer Society.

[17] Dekleva, S. (1992). *Delphi Study of Software Maintenance Problems*. IEEE Conference on Software Maintenance, ICSM 1992, pp. (10-17). Orlando, Florida, USA.

[18] El Emam, K., & Madhavji, N.H. (1995). *Measuring the success of requirements engineering processes.* In Requirements Engineering, Proceedings of the Second IEEE International Symposium, pp. (204-211).

[19] Feilkas, M., Ratiu, D., & Jurgens, E. (2009). *The loss of architectural knowledge during system evolution: An industrial case study*. 17th International Conference on Program comprehension, ICPC '09, pp. (188-197).

[20] Ferrari, R., & Madhavji, N.H. (2008). *Architecting-problems rooted in requirements*. ACM journal of Information and Software Technology archive, 50(1-2), pp. (53-66).

[21] Firesmith, D. (2007). *Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them.* Journal Of Object Technology, 6(1), pp. (17-33).

[22] Forward, A., & Lethbridge, T. C. (2002). *The Relevance of Software Documentation, Tools and Technologies: A Survey.* The ACM Symposium on Document Engineering, pp. (26-33). McLean, Vancouver, Canada.

[23] Gall, M., Bruegge, B., & Berenbach, B. (2006). *Towards a Framework for Real Time Requirements Elicitation.* First International Workshop on Multimedia Requirements Engineering, 2006. MERE '06, pp.(4-4). Minneapolis/St. Paul, Minnesota: IEEE Computer Society. 9

[24] George, B., Bohner, S. A., & Prieto-Diaz, R. (2004). *Software information leaks: A complexity perspective.* Ninth IEEE International Conference on Engineering Complex Computer Systems (ICECCS'04), pp. (239-248). Florence, Italy: IEEE Computer Society.

[25] Gross, A., Doerr, J. (2012). *What do software architects expect from requirements specifications? results of initial explorative studies*. IEEE First International Workshop **on** Twin Peaks of Requirements and Architecture, IEEE Software, pp. (41-45).

[26] Guion, L. (2002). *Triangulation: Establishing the Validity of Qualitative Studies.* University of Florida Extension: Institute of Food and Agricultural Sciences.

[27] Gumuskaya, H. (2005). Core Issues Affecting Software Architecture in Enterprise Projects. Proceedings of World Academy of Science, Engineering And Technology, volume 9, pp. (35-41)

[28] Hamilton, F., & Johnson, H. (1995). *An empirical study in using linked documentation to assist software maintenance.* International conference on Human-Computer Interaction, 1995, pp. (219-224).

[29] Hesse, M., & Paech, B. (2013). *Supporting the Collaborative Development of Requirements and Architecture Documentation*. IEEE Second International Workshop on Twin Peaks of Requirements and Architecture, IEEE Software, pp. (22-26). Rio de janeiro, Brazil.

[30] Host, M., Regnell, B., & Wohlin, C. (2000). *Using Students as Subjects – A Comparative Study of Students and Professionals in Lead-Time Impact Assessment.* Empirical Software Engineering, pp. (201-214).

[31] IEEE-std-1471-2000: Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE, http://standards.ieee.org/.

[32] IEEE-std-830-1998: Recommended practice for software requirements specifications. IEEE, http://standards.ieee.org/.

[33] Johnson, R. B., and Christensan, L. (2004). Educational Research: Quantitative, Qualitative and Mixed Approaches. Allyn & Bacon (2nd Ed.).

[34] Kamsties, E., Hormann, K., & Schlich, M. (1998*). Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions, and Technology Transfer*. International conference on European Industrial Requirements Engineering (CEIRE'98), pp. (30 41). London,UK.

[35] Kannenberg, A., Saiedian, H. (2009). *Why Software Requirements Traceability Remains a Challenge*. The Journal of Defense Software Engineering, pp (14-19).

[36] Kotonya, G., & Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques.* John Wiley & Sons.

[37] Koziolek, A. (2012). *Architecture-driven quality requirements prioritization*. IEEE First International Workshop on Twin Peaks of Requirements and Architecture, IEEE Software, pp. (15-19). Chicago, IL.

[38] Lamsweerde, A. (2000). *Handling Obstacles in Goal-Oriented Requirements Engineering*. IEEE Transactions on Software Engineering, 26(10), pp. (978-1004)

[39] Lee, S., & Rine, D. (2004). *Missing Requirements and Relationship Discovery through Proxy Viewpoints Model*. 19th annual ACM Symposium on Applied Computing, pp. (1513-1518). Nicosia, Cyprus.

[40] Lethbridge, T. C., Singer, J., & Forward, A. (2003). *How Software Engineers Use Documentation: The State of the Practice.* IEEE Software, *20* (6), pp. (35-39).

[41] Liang, P. , Avgeriou, P., & Keqing He. (2010). *Rationale management challenges in requirements engineering*. Third International Workshop on Managing Requirements Knowledge (MARK), pp. (16-21). Sydney, NSW.

[42] Menten, A., Scheibmayr, S., & Klimpke, L. (2010). *Using audio and collaboration technologies for distributed requirements elicitation and documentation.* Third international workshop on managing requirements knowledge (MaRK'10), pp. (51 – 59). Sydney, Australia.

[43] Miranskyy, A., Madhavji, N., Davison, M., & Reesor, M. (2005). *Modelling Assumptions and Requirements in the Context of Project Risk*. 13th IEEE International Conference on Requirements Engineering, IEEE software, pp. (471-472).

[44] Poort, E.R., Key, A., de With, P.H.N., & van Vliet, H. (2012). *Issues Dealing with Non Functional Requirements across the Contractual Divide*. International Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), pp. (315 319). Helsinki, Finland.

[45] Ralph, B., Lisa, M., & Alexa, S., (2006), Introduction to Applied Biostatistics, Thompson.

[46] *Requirements Tools*. (n.d.). Retrieved November 2009, from Volere Requirements Resources: http://www.volere.co.uk/tools.htm.

[47] *Research Methods Knowledge Base*. (n.d.). Retrieved November 2009, from Social Research Methods: http://www.socialresearchmethods.net/kb/.

[48] Rosenhainer, L. (2004). Identifying Crosscutting Concerns in Requirements Specifications, International journal of Softwaretechnik-Trends, 24(1), pp. (15-25).

[49] Runeson, P., (2003). *Using Students as Experiment Subjects – An Analysis on Graduate and Freshman Student Data.* Proceedings 7th International Conference on Empirical Assessment & Evaluation in Software Engineering, EASE'03.

[50] Thelin, Thomas, (2004). *Team-based fault content estimation in the software inspection process.* 26th International Conference on Software Engineering (ICSE 2004), pp. (263-272).

[51] Schneider, K., Stapel, K., & Knauss, E. (2008). *Beyond Documents: Visualizing Informal Communication.* Requirements Engineering Visualization, 2008, REV'08, pp. (31-40). Barcelona, Spain: IEEE Computer Society.

[52] Sinha, V., Sengupta, B., & Chandra, S. (2006). *Enabling Collaboration in Distributed Requirements Management*. IEEE Software , 23 (5), pp. (52-61).

[53] Solemon, B., Sahibuddin, S., & Ghani, A. A. (2008). *Requirements engineering problems in 63 software companies in Malaysia.* International Symposium on Information Technology, 2008. ITSim 2008, 4, pp. (1-6). Kuala Lumpur, Malaysia.

[54] Sommerville, I and Sawyer, P. (2000). *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons Ltd.

[55] StatisticalHelp from StatsDirect. (n.d.). Retrieved November 2009, from Stats Direct: Software to improve statistical practice: http://www.statsdirect.com/help/.

[56] Su, Chao-Tung., Dowming Yeh. (2012). *Software Architecture Recovery and Redocumentation Tool of a Hospital Information System*. International Conference on Computer and Communication Engineering (ICCCE), 2012, pp. (143-147).

[57] The Standish Group Report (CHAOS). (2003). Retrieved November 2013, from :http://www.projectsmart.co.uk/docs/chaos-report.pdf.

[58] Whalen, M. W., Murugesan, A., & Heimdahl, M. P. E. (2012). *Your What is My How: Why Requirements and Architectural Design Should Be Iterative*. First IEEE International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks), pp. (36–40).

[59] Wiegers, K. (2001). "Inspecting Requirements," *StickyMinds.com Weekly Column*, 30 July, 2001. Retrieved from: http://www.stickyminds.com.

[60] Yin, R. K. (2003). *Case Study Research: Design and Methods* (3rd Ed.). Sage Publications.

Appendix

Instruments design (Templates questionnaires)

We have used excel sheets to create different templates that we have used for data collection. We did ask several questions and there are around 23 columns in our excel sheet. Due to that we cannot represent it here as a tabular format, rather we are giving the sample questions that we have asked in different templates.

Updated requirements template (URT):

Purpose: The purpose of this template is to record updates to the original requirements.
Group Number: _____

Template Author: ONLY Entire group working together (names not needed).

Instructions: There are 17 columns in the template below.

Column B – Requirement ID.
Column C– The scope of the system to which the requirement applies.
Column D– Description for the requirement.
Column E– Priority of the requirement. (VH-very high,H-high,M-medium,L-low,VL-very low)
Column F– If any requirement depends on other(one/more) requirements then write down requirements ids on which it depends on.
Column G: Source of the requirement
Column H: Assumption, rationale behind the requirement
Column I – Different change types (add/delete/modify). Choose one option. List was given as follows:

Add one or more requirements
Delete one or more requirements
Modify specific requirement
Modify ATTRIBUTES of a requirement

Column J – Which architectural activity were you conducting when the need for the requirement-change occurred. Choose one option. List was given as follows:

Requirements Understanding
Feature Identification
Domain_analysis-Context diagram
Domain_analysis-Use case diagram
Domain_analysis-User scenarios
Modelling Quality Attribute Scenarios
Determine Architectural drivers
Determine Architectural patterns
Determine Architectural tactics
Overall_system_Architecting-Module level view
Overall_system_Architecting-Deployment view
Overall_system_Architecting-Component & Connector view
Overall_system_Architecting-Decomposition view
Module Interface definition
Documenting the architecture
Other

Column K – Write the requirement.

(NEW: write one or more requirements each with a unique Id;

DELETE: Insert the term "DELETE" in column 8 and colour the original requirement RED;

MODIFY: Insert the modified requirement in column 8 and colour the original requirement RED).

Column L– Rationale behind the change (application domain/requirements related issue/other). Choose an option.

Column M– Give details for the option selected in column 9.

Column N– If requirement is not changed but its attribute is changed (Priority, Rationale, Assumption etc.) then choose the changed attribute name from the list. List options was as follows:

Source of requirement
Priority
Dependency
Rationale
Assumption
Other

Column O– Give details reasoning for the attribute change.

Column P – Time spent in making the change to the requirement (includes discussing the idea of

change, drafting rough cut (preliminary) requirement, refining the requirement, etc.)

Column Q – Changed requirement was architecturally significant or not.(Yes/No)

Column R – Write details of the significance.

Column S–Changed requirement was Cross-cutting or not? Choose an option. (If a requirement is applicable to multiple components of the system then that requirement is known as a cross-cutting requirement).

Column T– Write details of the Cross-cutting.

Column U –If a NEW requirement is ADDED, indicate whether the requirement was IMPLICIT in the original document or was it completely NON-EXISTANT in the original document. (An IMPLICIT requirement is one that is ASSUMED but not documented in the original document; A NON-EXISTANT requirement simply did not exist --even IMPLICITLY-- in the original document). Choose an option.

Column V– A Non-functional requirement is the same thing as Quality requirement. Choose an option.

Requirements Missing Information Template (RMIT)

Purpose: The purpose of this template is to record the requirements-related MISSING information. If, during architecting, you recognise the need for requirements information that is not in the requirements document then such information must be captured in this template.

Group Name: ___

Template Author: (write "X" --choose one)

_____Individual working alone (name): _____
____Subgroup working jointly (member names): __
_____Entire group working together (names not needed)

Who should use it and WHEN:
For EVERY session:
• Every individual in the group must have his/her own RMIT template.
• Any subgroup working jointly should have its own RMIT template.
• The entire group working together should have its own RMIT template.

Instructions: There are 11 columns in the template below.

Column B – Fill in the date and time for the identified missing information

Column C – Is there any information you were expecting in the requirements document but did not find it there? Choose an option.

Column D – What kind of information was you looking for and did not find it in the requirements document? Explain details for column C.

Column E – Why did you expect that this information should be in the requirements document? Give reason.

Column F– In which architecting activity was the need felt for the missing requirements information? Choose an option. We have following SA activities listed:

Requirement Understanding
Identifying features
Domain_analysis-context diagram
Domain_analysis-Use case diagram
Domain_analysis-User scenarios
System_decomposition-Quality Scenario
System_decomposition-Architectural drivers
System_decomposition-Architectural patterns
System_decomposition-Architectural tactics
Overall_system_Architecting-Module level view
Overall_system_Architecting-Deployment view
Overall_system_Architecting-Component & Connector view
Module Interface definition
Documenting the architecture-artefact
Documenting the architecture-Textual description
Documenting the architecture-Architecture background analysis
Other

Column G – In which way was the missing information relevant to the chosen architecting activity? Choose one or more items. We have given the following options:

Help in the choice of architecturally sensitive requirements
Help in modelling Quality Attribute Scenarios

Help in the choice or design of Tactics
Help in the choice or design of Architectural patterns
Help in the analysis of sensitivity points
Help in the analysis of trade off points
Help in modelling architectural views
Help in specifying component interfaces
Help in creating context diagrams
Help in deploying the architecture on hardware elements
Help in analysing the quality of the architecture
Other

Column H– If you created the missing information (during architecting) then identify all the Requirement Engineering activities you conducted to create that information. Choose one or more items. The list was given as follows:

Elicitation & Negotiation
Analysis of requirements
Modelling requirements
Prioritising requirements
Documenting requirements

Column I–If you did not create any information then give explanation for the work around.

Column J –Time in Minutes; Number of people.

Column K– How critical is the missing information for architecting purposes? (Very Critical-5, Medium Critical-4,Low critical -3,Very low-2,Not critical-1). Choose an option.

Column L- Explain details for column K.

TLT (Time log template):

Purpose: The purpose of this template is to record time spent on tasks.

Group Number: _____

Template Author: (Please place an 'X' next to one)
____ Individual working alone (name): _____
___ Subgroup working jointly (member names): _____
____ Entire group working together (names not needed).

Who should use it and WHEN:

For **EVERY working session there must be a TLT template capturing time log for this session.**

- **Every individual** in the group must have his/her own TLT template.
- **Any subgroup** working jointly should have its own TLT template.
- **The entire group** working together should have its own TLT template.

Instructions: There are six columns in the template below.

1st column – This is where the date on which a given task begins or continues.
2nd column – The start time for the task is recorded.
3rd column – The stop time for the task is recorded.
4th column – Any interruptions, in minutes, are recorded.
5th column – A brief summary of the task that is being carried out is recorded here.
6th column – Precise references to all documents (e.g., reports, section in the report, Figure number, book chapter, etc.) that were involved in this task.

Date	Start Time	Stop Time	Interruption Time (min.)	Task summary	Ref.

Curriculum Vitae

Name: Md Rounok Salehin

Post-Secondary Education and Degree:

The University of Western Ontario
London, Ontario, Canada
2012-13
M.Sc., Computer Science

Islamic University of Technology
Dhaka, Bangladesh
2007-2010
B.Sc in Computer Science

Related Working Experience:

Teaching and Research Assistant
The University of Western Ontario
September 2012- December 2013

Lecturer
Stamford University Bangladesh
March 2011-August 2012

Software developer (Team member)
British American Tobacco, Bangladesh
November 2010- February 2011

System Engineer (Intern)
Networks division
Grameenphone Ltd
October 2009-January 2010