Master Thesis
Software Engineering
Thesis no: MSE-2009:27
November 2009

Minimizing Defects Originating from Elicitation, Analysis and Negotiation (E and A&N) Phase in Bespoke Requirements Engineering

Israr Ahmed
Shahid Nadeem

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Contact Information:
Author(s):

Israr Ahmed
E-mail: sh_sheikh03@yahoo.com

Shahid Nadeem
E-mail: shahid.nadeim@gmail.com

University advisor(s):
Tony Gorschek
Department of System and Software Engineering

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet `: www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

ABSTRACT

Defect prevention (DP) in early stages of software development life cycle (SDLC) is very cost effective than in later stages. The requirements elicitation and analysis & negotiation (E and A&N) phases in requirements engineering (RE) process are very critical and are major source of requirements defects. A poor E and A&N process may lead to a software requirements specifications (SRS) full of defects like missing, ambiguous, inconsistent, misunderstood, and incomplete requirements. If these defects are identified and fixed in later stages of SDLC then they could cause major rework by spending extra cost and effort. Organizations are spending about half of their total project budget on avoidable rework and majority of defects originate from RE activities. This study is an attempt to prevent requirements level defects from penetrates into later stages of SDLC. For this purpose empirical and literature studies are presented in this thesis. The empirical study is carried out with the help of six companies from Pakistan & Sweden by conducting interviews and literature study is done by using literature reviews. This study explores the most common requirements defect types, their reasons, severity level of defects (i.e. major or minor), DP techniques (DPTs) & methods, defect identification techniques that have been using in software development industry and problems in these DPTs. This study also describes possible major differences between Swedish and Pakistani software companies in terms of defect types and rate of defects originating from E and A&N phases. On the bases of study results, some solutions have been proposed to prevent requirements defects during the RE process. In this way we can minimize defects originating from E and A&N phases of RE in the bespoke requirements engineering (BESRE).

Keywords: E and A&N, RE, SDLC, BESRE, most common requirements defect types, DPTs, empirical study, literature study, interviews, literature reviews, requirements defect taxonomy, adoptability

ACKNOWLEDGEMENT

Foremost, we would like to express our utmost gratitude to our supervisor Dr. Tony Gorschek for providing us consistent and valuable support in research. We thank to him for giving us precious time and attentions whenever required. A part from these, we also admire his interest in research with numerous discussions, constructive comments, and beneficial suggestions which not only improved this work but also greatly helped us to utilize our effort in the right direction.

It is our pleasure to admire all those interviewees from Pakistan and Sweden which gave us appropriate time and showed immense interest in research topic. We appreciate them for giving us useful information in data collection process. Without their involvement we were unable to get good results.

We greatly acknowledge the Software Engineering Department at BTH for providing us useful information and guidelines in conducting a good research. We also admire Library staff for providing us technical resources for the thesis. They were helpful for providing us an environment to access online databases regarding research resources and related materials.

We are also greatly indebted to our friends for their continued love and support. Finally, we intensely appreciate the moral support from our parents. Their advices, love and encouragement had really made this thesis possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The RE process is the initial important phase of SDLC that is used to discover and develop requirements for a system [4]. In general, this process is employed to explore the customer's needs, analyze those needs to find problems, achieve feasibility, negotiate appropriate solution, specify the solution, validate real customer's needs, and manages the requirements [17]. From preceding definition it is clear that the RE process consists of requirements elicitation, analysis & negotiation, specification, validation and requirements management activities. The requirements elicitation phase involves communication with system stakeholders i.e. customer and end-users etc to identify system requirements. It is the most difficult, error prone, most critical, and most communication intensive aspect [4].

The requirements analysis and negotiation (A&N) is the process of analyzing the system requirements to identify conflicting, unnecessary and incomplete requirements and to negotiate them with stakeholders to reach an agreement. The E and A&N phases are very tightly coupled with each other. [7] A poor E and A&N process may result an SRS full of missing, ambiguous, overlapping, incomplete, inconsistence, conflicting, infeasible, unprioritize, unverifiable and unrealistic requirements [4, 7]. Software industry has been trying to develop a quality SRS by using different techniques, and methods but besides of these attempts more than 50 to 70 percent defects are originating from the RE process. Defect correction in later stages of SDLC is more expensive, time consuming and hard to fix than in early stages [2, 3, 51]. It is also noted that software projects spend about 40 to 50 percent of their total effort on avoidable rework. So there is need to minimize these requirements level defects as early as possible to avoid their later consequences [3, 4]. Wesley in 2001 has explained that from 80 percent defects associated with requirements, 49 percent are due to incorrect assumptions, 29 percent result by omitted requirements and 56 percent of the defects are due to poor communication between user and analyst [6]. It means major source of requirements defects are E and A&N phases in the RE process and action should be taken on these phases to prevent defects.

To reduce rework generated as a consequences of requirements level defects, the authors will try to minimize avoidable rework in later stages of SDLC. This will be done by identifying most common requirements defects types reported from literature research (literature review) and industrial research (interviews). The authors will then look for reasons for these defects, DPTs and methods along with their strengths and weaknesses. Based on this information, the authors will propose appropriate solution in the form of list of recommendations, or modification in DPTs or methods to achieve research goals.

2 BACKGROUND

The RE process is the structure set of activities involves in discovering, documenting and managing the requirements for a system [7, 18]. In general, it provides the understanding of what the customers want, analyzing their needs, assessing feasibility, negotiating a reasonable solution, specifying the solution, validating the specification with respect to customers, and managing the requirements as they are transformed into an operational system [17]. The RE process has two main components 1) Requirements Development and 2) Requirements Management [4]. Requirements development process includes requirements elicitation, analysis and negotiation, specification and validation activities. A brief detail of these activities is given below in figure 1.

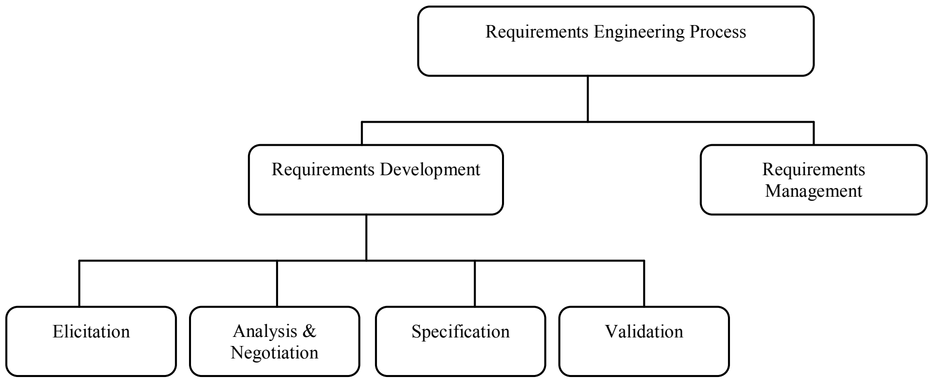


Figure 1: RE process activities [4]

The RE process starts with the requirements elicitation activity which involves communication with system stakeholders i.e. customer and end-users etc to identify system requirements. According to Gorschek [18], "the requirements elicitation is the process of discovering the requirements for a system by communication with the stakeholder and through the observation of them in their domain". Christel and Kang have stated that deficiencies in this phase can directly lead to problem of scope, understandability of customer's need and requirements volatility [17]. Elicitation phase covers all four levels of requirements like business, user, functional and system requirements [4]. These four levels of requirements are discovered by having discussion with stakeholders, system documentation, domain knowledge and market studies [7]. The requirements elicitation is performed using different elicitation techniques. Commonly used techniques at this phase are interviews, surveys, scenarios, observation, group elicitation techniques which includes brainstorming, Joint Application Design (JAD) workshops, group interviews and brainstorming sessions detail of these can be find in [7, 17, 18].

The requirements A&N is the process of analyzing the system requirements in order to identify conflicting, unnecessary and incomplete requirements and to negotiate them with stakeholders to reach an agreement on changes that satisfy all stakeholders' needs. A cross check is performed among requirements to identify conflicts. Requirements prioritization is the part of requirements negotiation (RN) by communicating with different stakeholders which put priorities on different system requirements [7]. It helps in finding the most important requirements for a system. A set of candidate requirements are given to the prioritization process and after having negotiation with customer, most important requirements are selected [4]. AHP, 100-point method and planning game are examples of prioritization techniques.

E and A&N are most erroneous phase of the RE process. Different techniques and methods have been using in accomplishment of this phase such as interview, brainstorming, QFD, JAD etc. Requirements analysis gets inputs from the elicitation phase and focuses on requirements necessity checking, consistency and incompleteness checking, overlapping, conflicting requirements, and requirements feasibility checking as shown in figure 2 [7].

RN phase gets activate when conflicts were found during requirements analysis. These conflicts are resolved by discussing it with customers. It is very time consuming phase and conflicts are not settled down perfectly. RN tries to establish a compromise between customers or users and final requirements are always a compromised set of requirements. This compromise is based on needs of the organization in general; budget and schedule for the system development, the specific requirements of different stakeholders, design and implementation constraints. Whenever requirements analyst discovers a conflict or a problem, the E and A&N phases are re activated and to get more information about that particular conflict or problems and try to resolve it. Here we can say that E and A&N processes are segment in a spiral. Activities in E and A&N are described in more detail in figure 2 from which it is clear that E and A&N are closely related and linked process. [7]

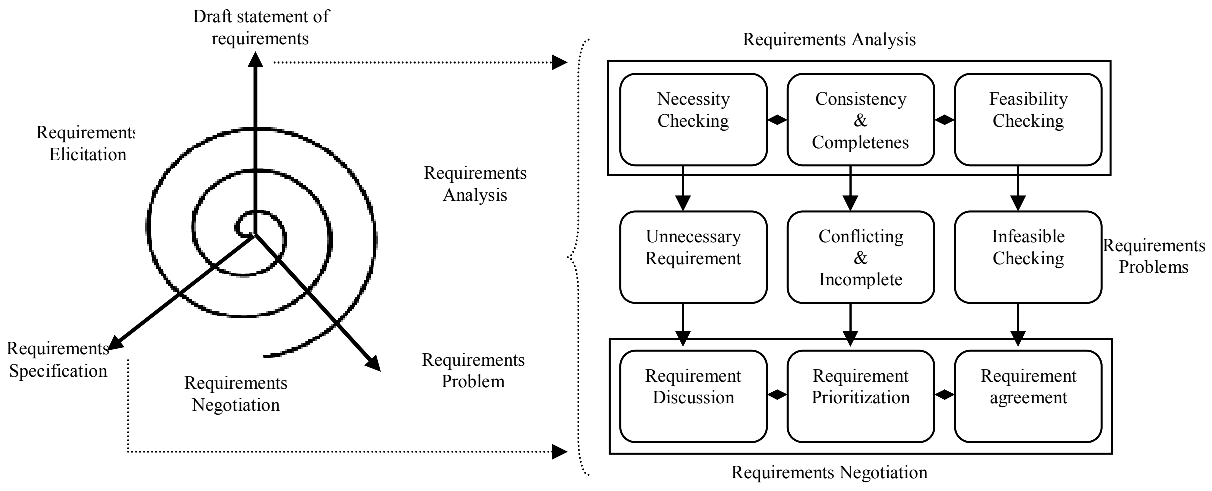


Figure 2: Requirements E and A&N spiral [7]

The output of this spiral would be refined and agreed set of requirements [7]. A bad E and A&N process may lead to an SRS full of missing, ambiguous, overlapping, incomplete, inconsistence, conflicting, infeasible, unprioritize, unverifiable and unrealistic requirements [4, 7]. Industry has been using a lot of techniques, tools and methods to develop refined and correct requirements for a system but in addition to these attempts more that 50 to 70 percent defects have been reported from the RE process that have been causing major rework in later stages of SDLC [3, 4]. So there is need to minimize these requirements level defects as early as possible to avoid their later consequences.

Requirements specification phase has gained special importance for last few years. The objective of this phase is to create requirements specifications so accurate that it can provide functionalities or system behavior that the user wants actually [42]. In other words Requirements specification is the official statement of agreed set of system requirements in such a way that is understandable by customer, end-users and people involved in software development i.e. tester, software developer, designer etc. [7]. Kelly et al. has said that there are more defects found in SRS than any other document developed during SDLC [50].

Requirements validation is the process of identifying problems in requirements specification and to ensure that requirements are realized in the right way according to the customer. Inspection and reviews can be used to identify defects in SRS [7, 17, 18, 19, 22]. Validation process isn't just a single unique phase that is performed after gathering and documenting all the requirements. Some validation activities, such as incremental reviews of the growing SRS, are carried out throughout the iterative elicitation, analysis, and specification processes [7].

Requirements management is the process of managing changes in requirements due to development of customer awareness during system evolution. The principle concerns of requirements management are; 1) managing changes to the agreed requirements, 2) managing the relationships between requirements, 3) managing the dependencies between documents produced during system development process [7].

DP is the process of finding the root causes of defects and prevents them from reoccurring [1]. Defect correction in later stages of SDLC is more expensive, time consuming and hard to fix than in early stages [2, 3, 51]. Fairley has presented data that shows that it is 5 times more costly to correct a fault at the design stage than during initial requirements, 10 times more costly to correct it during coding, 20 to 50 times more costly to correct it at acceptance testing, and 100 to 200 times more costly to correct that problem during actual operation (a variation on the theme "pay me now or pay me later") [20]. Karl E. Wiegers seems to be agreeing with Fairley's data presentation as he has shown in figure in 3. Karl E. Wiegers has observed that preventing requirements defects or identifying them in early stages provides huge positive effect in reducing rework in later stages [4].

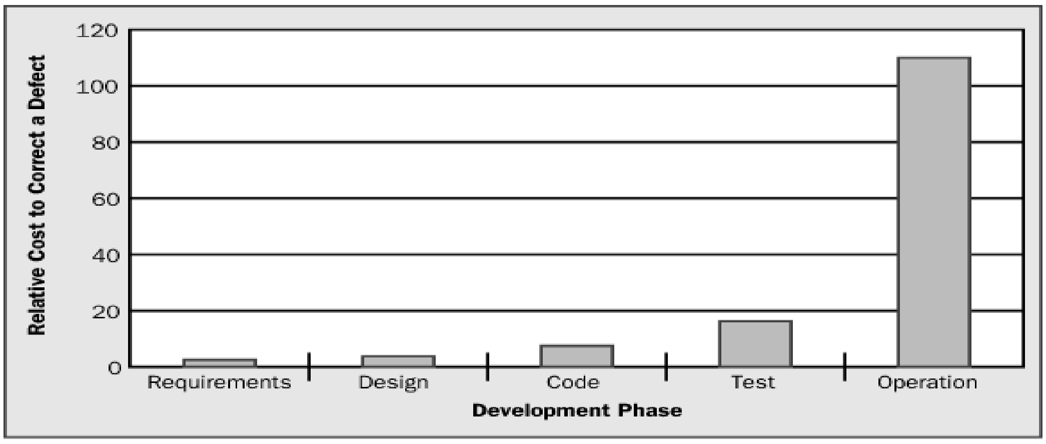


Figure 3: Requirements defect correction in different phases and relative cost [4]

It is found that most of the organizations use a reactive strategy (find and fix after delivery) to deal with them instead of using a proactive approach (DP) [1, 36]. This reactive strategy is often 100 times more expensive than finding and fixing the defects during requirements [2]. Software projects spend about 40 to 50 percent of their total effort on avoidable rework [3, 4]. The earliest solution is DP to optimize the development and rework cost [4, 5]. Karl Wiegers has proposed that more than 40 to 70 percent of defects found in the RE process which causes a lot of rework in later stages of SDLC [4]. Almost 70 percent of the system errors are due to inadequate system specification, lack of user inputs and changing customer's requirements [6]. Wesley in 2001 proposed that 80 percent of the defects in developed software originate from requirements due to incorrect assumptions (49%), omitted requirements (29%), inconsistent (13%) and ambiguous requirements (5%). 56 percent of the defects are due to poor communication between user and analyst in defining requirements resulting 82 percent of rework in later stages of SDLC [6]. That's why there is a need to converge the focus from SDLC to RE activities in the context of DP.

In the BESRE, as mentioned above that out of 80 percent defects 49 percent are due to incorrect assumption, 29 percent result by omitted requirements and 56 percent of the defects are due to poor communication between user and analyst (Wesley et. al, 2001). This demonstrates that the major sources of defects among the BESRE activities are E and A&N phases, so action should be taken to prevent defects in these activities to avoid their later consequences. So we converge our focus to E and A&N phases of the BESRE process as shown in figure 4. Further in requirements development, the requirements elicitation is the most difficult, error prone, most critical, and most communication intensive aspect [4].

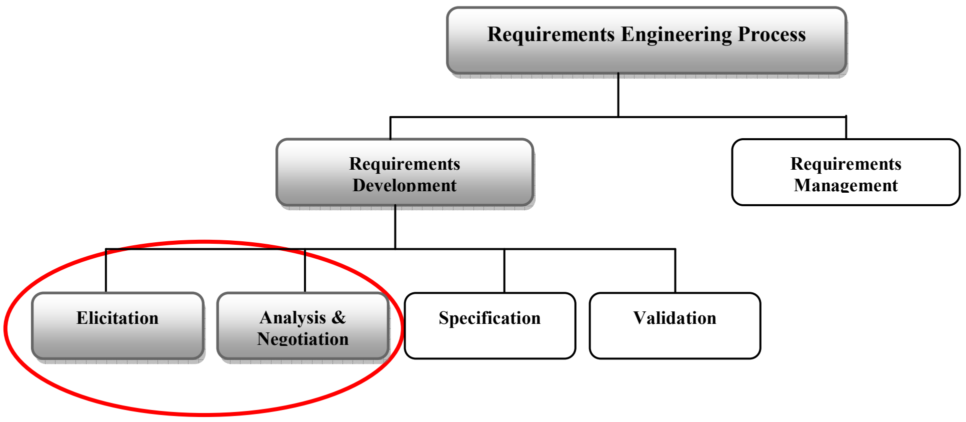


Figure 4: Our focused RE activities

It is not just to ask what the customer's needs are but it has multiple dimensions that cause different problems for requirements analyst. These problems include 1) application knowledge doesn't exist in one place but it is found in multiple places like textbooks, working manuals and in people's head who has been working in that area 2) organizational issues and political factors may affect the collected requirements 3) stakeholders often don't know what they want for their system. They don't realize the importance of giving complete and detailed requirements 4) RE teams do not want to spend more time in the RE process. [7]

RE phases have many problems like defining the system scope, misunderstanding among different communities (such as customers and developers) affected by the development of a given system, and problem of volatility in requirements. By improving the requirements elicitation process, software engineering process might be improved. In this way good system requirements specification could be developed and ultimately a much better system will be developed. An improved and quality requirements elicitation process also guarantees reduction in requirements errors that cause tremendous rework when those are happened in later stages of SDLC. The IEEE Guide to SRS has defined some attributes for a good SRS. These attributes are unambiguous, complete, verifiable, consistent, modifiable and traceable. A good elicitation process helps in the development of SRS with above attributes. On the other hand a bad elicitation process will develop an SRS that will be full of incomplete, unverifiable, inconsistent, non modifiable and untraceable requirements. [77] Requirements E and A&N are very closely linked processes. Software requirements A&N confirms agreed set of requirements that must be complete and consistent [7]. From this discussion, it seems that poor elicitation process leads to poor SRS which ultimately leads to erroneous software system and that's why our focused is on E and A&N phases of RE.

2.1 Related Work

In order to deal with requirements level defects, if defect information (most common defect types, defect sources, etc) from past project is considered during review then it is easy to overcome their consequences in later stages. Some Methods use defect information to deal with defects which are defect-causal analysis is considered to be a low cast technique for minimizing defects in software developed by IBM in 90's and focus on learning from defects

[36]. It is a team-based approach (performed by a team) that focus on identifying classes of defects and than proposes actions to deal with them rather than correcting individual defects [21, 36]. Software bugs analysis also aims at determining the sources of defects [21]. Root cause analysis focuses to identify the actual causes of defects but it is time consuming, require experience and effective if there are large number of defects [21]. Error abstraction process analyzes the group of related defects to determine their causes based on inspector experience [21].

DP process is a proactive strategy that uses defect casual analysis to identify causes of defect and propose preventive actions to avoid those defects [21]. There are many DP approaches discussed in [21, 23] most common of which are Cleanroom, Join Application Design (JAD), quality function deployment (QFD) etc.

Current research regarding DP is focused on SDLC using various mechanisms [8, 9]. Quality standards like ISO, CMMI etc also propose actions to deal with DP such as Casual Analysis and Resolution, Corrective and Preventive actions etc [9]. They focus on explaining things in term of "what" instead of "how", which make them less effective. Requirements processes in these standards are less supported, less focused and less mature than other software processes in SDLC [9]. Techniques like inspection, prototyping and reviews etc are also widely used to detect defects in requirements [4]. Most of the requirements reading techniques (such as ad hoc, checklist, defect based reading) do not pay attention to particular aspects of the SRS and put all requirements information on the same level of importance. In this way the identification of all types of defects in the entire document becomes ineffective [22]. Methodology such as Cleanroom focus on formal specification and verification, QFD focus on customers and design parameters requirements, and JAD sessions are considered to be the most effective techniques to deal with early defects in requirement and design but they are costly and time taking approaches [10]. Schneider, Johnny Martin and W.T. Tsai have conducted an experiment to identify requirements level defects by inspecting SRS. They noted that if only one team is assigned for the inspection of SRS then only 27% defects were caught. When they used N-fold inspection technique and appointed ten teams then defect detection rate raised up to 80% [19]. It means ten times more resources are required to catch 80% requirements level defects. But it is realty that most of the organizations believe in finding and fixing defects in later stages of SDLC [1] and don't want to appoint ten teams for the inspection of requirements. Other different approaches are described in [11, 23, 31] to deal with defects a part from those describes above which are risk analysis, Personal Software Process (PSP), scenarios and focus group with users, etc. The continued paragraph reflects that there are a lot of DP methods, techniques and approaches in practice, but still more than 40 to 70 percent defects are originating from RE phases [4] and organizations are spending 40 to 50 percent of total effort over avoidable rework [3].

Research findings in the field of DP focused more on work product instead of process in the context of the BESRE [5, 11, 12]. As SRS is the primary work product at requirements level so authors will focus on SRS instead of RE process. Defects found in work products are mostly based on Beizer's Taxonomy proposed by Beizer in 1990 which act like a classification of predefined defect types from past projects [31, 34]. It is found in literature that most of the organizations develop their own defect taxonomies because defect data might be heterogeneous data that comes from different sources, at different periods of time, in different formats, have different terminologies and different problem needs of an organization [26, 31, 34]. Commonly used taxonomies are Hewlett Packard Defect Classification Scheme (HP-DCS) used for SDLC with limited number of defect modes (represent the nature of defect types whether it is missing, unclear or incomplete, etc) for defect types [76] that are difficult to use for requirements level defects. IEEE provides a standard classification mechanism for anomalies (IEEE use a common term "anomaly" for all type of problem, errors, defects, fault, and failures etc) [75]. NASA developed a taxonomy specific to requirements for its projects which is simple but does not contain enough detail for adoption

[73]. Boris Beizer in 1990s proposed detailed bug taxonomy for SDLC which contain enough detail to classify defects [32].

To deal with the problems and to reduce rework generated as a consequence of requirements level defects, the authors will try to reduce avoidable rework in later stages of SDLC by identifying most common defect types and reasons for defects using both academia and industry as sources. Then we will find how much rework would be done based on probability if a defect (from a defect type) would be identified and fixed in later stages of SDLC (i.e. design, implementation). We will then look for existing DPTs that are mostly used to handle these defects. On the basis of problems found in DPT, preventive action will be taken to minimize these defects so that they can cause little rework in later stages of SDLC. These DP actions might be change in existing DPT, enhancement in DPT, creating a new DPT, or a list of recommendations.

3 RESEARCH DESIGN

3.1 Aims and Objectives

The aim of this thesis to reduce the rework in later stages of SDLC that is caused by requirements defects originating from E and A & N phase in the BESRE;

1) Identify major defects types and reasons for defects related to E and A&N phases in the BESRE found in academia and industry
2) Analyze the defects types in order to find their impact (in the form of rework).
3) Analyzing differences if any in defect types between Swedish and Pakistani companies
4) Finding problems in DPT connected with defect types.
5) Propose preventive action based on defect information.
6) Reducing avoidable rework in the form of defect correction

3.2 Research Questions

1. What are requirements level defect types and reasons for defects related to E and A&N phases that can cause major rework in later stages of SDLC?

   1.1. What are the most common defects types reported by research based on E and A&N?

   Description
   Poor E and A&N phases in RE result different types of defects that cause major rework in later stages of SDLC. We want to find these defects types and their causes that have been reported by different industrial case studies in literature and academic research (articles, books).

   1.2. What are the most common defects types and reasons for defects originating from E and A&N as reported from Swedish and Pakistani software companies respectively?

   Description
   We want to conduct an empirical study for defects originating from E and A&N and want to know what common types of defects are there and what are their causes reported by current software development industry. For this purpose three Pakistani and three Swedish companies have been selected.

   1.3. Find possible rework caused by each defect?

   Description
   In this question We want to know that what types of defects cause major rework if it would be identified and fixed in later stages of SDLC. All types of defects would not be of the same importance. There must be some types of defects that are of minor importance with respect to rework. We will consider only those types of defects that cause major rework when they occur in later stages of SDLC.

2. Are there any major differences between Swedish and Pakistani companies in terms of types and rate of defects originating from E and A&N phase?

   Description
   There might be some differences in terms of defect types originating from E and A&N phase among Pakistani and Swedish companies due to difference in RE practices. The rate of origination of defect from E and A&N phase might be different among Pakistani and Swedish software development companies. There might be some similarities among both Pakistani and Swedish companies in terms of types and rate of defect originating from E and A&N phase.

3. What DPT is associated with each defect type that is being practicing in industry based on E and A&N?
   3.1. What is appropriate DPT for each defect type?

   Description
   There must be some DPTs to prevent defects originating from E and A&N. We want to know about these DPTs so that we can have a critical overview of these DPTs.

   3.2. What are problem in existing DPT (s)?

   Description
   Since We have mentioned above that more than 40 to 70 percent defects are originating from software requirements, so there must be some weaknesses or problems in current DPTs practicing in software development industry. We want to know these problems so that We can take preventive actions in the form of change in existing techniques, create a new technique or give some recommendations.

4. How can we remove the problems in DPT and make them more efficient to handle defects that cause major rework?

   Description
   After finding problems in existing DPTs practicing in software industries, how We will remove these problems? It might be change in existing DPT or creation of new DPT or We can give a list of recommendations.

5. To what extent the prevention actions are valid?

   Description
   In the end We have to validate our preventive actions as mentioned in description of RQ3.2

3.3 Expected Outcomes

The outcomes of our thesis will be;

1) Understandings of problems in E and A&N phase that causes defects
2) Understanding of DPT (s) suitable to deal with defects in these phases
3) Understanding of differences in defect types and rate of defects in Pakistan and Sweden industry in E and A&N phase
4) Preventive actions to improve the E and A&N phase

3.4 Research Methodology

The research questions that we have prepared to support the objectives of our thesis, need different types of research methodologies. The big picture of these research methodologies seem to be qualitative. The qualitative approach is used and is consistent with the nature of current study because of two things. First, it is a subjective approach which focuses on finding questions in term of “what”, “why” and “how” and second, it helps in understanding the things in their environment in which they operate concerning their social aspects [13]. The study comprises of literature review and interviews as sources of data collection.

Qualitative literature review will be conducted for the proposed study because it provides solid base to know the current status of the body of knowledge (what is already known (done) and what is need to know i.e. new research) regarding related research field [14]. We will follow Concept-centric approach that will tie literature with current study based on continuous focus on one important question during reviewing literature and writing the literature review. This important question is ‘how is the work presented in the article I read related to my study?

Data will be collected from literature reviews using literature sources based on Creswell priority list e.g. journals articles, books and conferences [16]. Data will be then processed to extract useful information by following data processing steps proposed by levy 1) know the material, 2) comprehend the material, 3) apply, 4) analyze by comparison, separation or making connection of cited information with respect to our research question and 5) evaluate [14].

The purpose of conducting interviews is to get current knowledge being practice in industry. It will provide more information from industrial expert in term of current practices, techniques related to E, A & N, common defects & defect types, defect sources, preventive actions and other related issues. Semi structure interviews (open-ended) will be use as qualitative approach to elicit information from industry because it provides a freedom of information to the participant involved [14]. Although this approach have risk of having inadequate and diverse information [15] but it is useful in this situation because of the interviewer's minute industry experience. Meeting will be arranged in advance and interviews will be conducted by using both face-to-face (in-person) and through phone calls options. During each meeting data will be recorded using the tape recorder and handwritten notes. Data will be then transcribed in to text form using MS word for analysis. This data will be analyzed through brainstorming and discussion which help to understand and interpret the data to find the target research questions. We have given in detail the appropriate research methodology for each research questions and corresponding objective and expected outcome achievement. This is shown in the table 1 given below.

Table 1: Relationship among research questions, research methodology, research objectives and outcomes

Research Question (RQ)	Research Methodology	Objective Achieved	Expected Outcomes
RQ 1	Literature Review, Interviews, We will create a defect taxonomy based on recommendations from literature.	1	1
RQ 1.1	Literature Review	1	1
RQ1.2	Interviews	1	1
RQ1.3	We will conduct interviews with experts (analyst, developer etc). He will show by his experience that what level of rework a particular defect can cause. This level may be Low, Medium, or High. In this way We will select defects that would cause High rework in later stage.	2	1
RQ2	Interviews will be conducted to experts both in Pakistani and Swedish companies	3	1
RQ3	We will conduct interview with experts and will ask about the DPTs and defect identification techniques that they are practicing in their company against defect they find. We will also do Literature Review.	4	2
RQ3.1	Literature Review, Interviews	4	2
RQ3.2	Literature Review	4	2
RQ4	On the bases of RQ3.2 We will take preventive actions. It will be done by consulting literature and our own creative ability.	5	3
RQ5	We will conduct interviews with experts to validate our preventive actions against problems in DPTs.	6	3

4 STUDY DESIGN

The proposed research study aims to find requirements level defects and their reasons through data collection using literature review from academia to learn from research carried out so far along with industrial experience that focus on what is happening in industry in practically. The study mainly focuses on finding requirements level defects types, their reasons, severity level with respect to rework, DPTs & DP methods, and problems in DPTs or DP methods. This information is achieved through data collection using literature review from academia and software development industry that focus on RQs described in section 3. This section contains how the study is planned, designed, and executed exactly in order to achieve RQs. The detail of this design is given below.

4.1 Literature Review Design

The literature part of study will contribute to figure out requirements level defect types that cause major rework in later stages of SDLC along with commonly used defect taxonomies and DPT's. The authors did literature review to find out above mentioned information. This section explains the literature review design which describes what kinds of resources are used for research articles, what selection criteria for articles were, how useful information was extracted from articles, and how exactly the literature review design was executed.

4.1.1 Literature Review Approach

We will follow Concept-centric approach that will tie literature with current study based on continuous focus of question that the author will keep in mind during reviewing literature and writing the literature review. This important question is 'how is the work presented in the article I read related to my study?

4.1.2 Literature Review Resources

Data is collected from literature reviews using literature sources based on priority list proposed by Creswell e.g. journals articles, books and conferences [16]. The literature resources consist of RE conferences and following electronic databases

- IEEE Xplorer
- ACM Digital Library
- Springer Links
- Science Direct (Elsevier)
- Engineering Village (Compendex, Inspec)
- Wiley Inter Science

4.1.3 Selection of Articles

The papers are selected from literature sources like ACM, IEEE and Springer Link etc. This selection is based on search criteria comprises of

- The articles should be peer reviewed
- The article can be a literature review, systematic review, an experiment, case study or survey report.
- The article should discuss RE level defects fully (specific to RE) or partially (discussed in SDLC)
- The article should discuss DPTs applied at RE level or they can be life cycle oriented techniques or methods.
- The articles should discuss software defect taxonomies, requirements error taxonomies or software defect taxonomies.
- The articles should discuss the most recent research related to a particular topic of interest.

4.1.4 Data Processing

Data will be then processed to extract useful information by following data processing steps proposed by levy [14].

1). **Know the material:** it means researchers have to demonstrate that they have thoroughly read the article and have extracted meaningful information from it. They can do it by performing four activities like listing, defining, describing and identifying the things.

2). **Comprehend the material:** after knowing the material the researcher should be able to summarize, interpret, differentiate, and contrast the concepts. It means researcher not only just repeat what was included in the article but they also know the true meanings and significance what was reported by the article.

3). **Apply:** it deals with identification of major concepts of the study and placing them in correct categories.

4). **Analysis:** analyze by comparison, separation or making connection of cited information with respect to research question

5). **Evaluate:** evaluation in literature review demands clearly distinguish the opinions, theories, and empirically extracted facts.

4.1.5 Exact Execution of Design

Concept-centric approach was strictly followed by the authors during reviewing literature. The authors visited several hundreds of research articles that were supposed to be related to their RQ but only 82 articles were selected according to criteria to the selection of research articles. These selected articles are firmly related to RQs that depend on qualitative review of literature.

Since the authors had enough time and full access to all literature resources (mentioned above), so they followed all resources with good search queries to make literature review effective and fruitful. On the other hand the data processing was not an easy task for the authors. They tried their best to follow data processing steps that were recommended by Levy (mentioned in section 4.1.4)

The analysis of literature study results was done in the same manner as it was done on industrial study results (see section 4.2.8)

4.2 Qualitative Interview Design

Qualitative interviews will be conducted for data collection from software expert in industry. Interview is an effective way of eliciting and learning information about research topic from interviewee experience in their domain [54]. According to Kahn and Cannel, interview is a meaningful and purposeful discussion between two or more persons [55]. During interview, interviewee is the main source of information who shares his experience, believe, opinion, and personal feeling about research topic based on questions posed by interviewer [54]. Generally, interview is conducted in two ways; one is using fact-to-face or in-person interview and second is conducted through telephone or call over internet using various sources e.g. Skype or messenger etc. [16]. For current study, authors will conduct interviews in six software industries both in Pakistan and Sweden (three from each) to know about requirements level defects and techniques to deal with them. Before conducting qualitative interviews, authors have designed the interview part in the following way.

4.2.1 Interview Goal

The goal of conducting interviews in industry is to know requirements level defects that originated from E and A&N phases of RE and causes a lot of rework if they are reported by customer after software put into operation. The goal of interview will be achieved through research questions related to qualitative interview described in table 1. The interview part of research study mainly focus on finding most common defect types, reasons for those defects, defect rates, expected rework in case the defects occur and technique (s) to deal with them. To achieve interview goal, the plan of actions to accomplish thesis goals consists of following activities.

4.2.2 Subject Selection

Interview data will be collection from software industry both in Pakistan and Sweden. Three companies from each country will be selected in this regard. The purpose of selecting organizations from two countries help to understand the variation of domain and environment and other aspects as well specifically related to RE. To get useful and correct information about requirements level defects originated from E and A&N phases, it is necessary to select those people for interview which are involved in RE activities. These people can be requirements analyst, developer, tester or product manager. They have a deep understanding of the RE process and work product. Tester usually generate test cases during requirements validation process to determine the correctness of requirements, requirements analyst on the other hand is involved in E and A&N process. To implement requirements, developer and analyst must have same opinion about agreed set of requirements. Product manager has the vivid picture of ongoing projects in the organization and he is familiar with the defects during in-process and product after deployment. He also knows the policies and organization resources to deal with on-going project problems. As a whole, the subject for interview is selected on following basics;

1) Subject should be directly involved in the RE process
2) Subject could be from quality assurance department and have complete understanding of requirements related defects
3) Subject should have three to five years of experience and should hold a senior position in organization

4.2.3 Interview Technique

According to [54] [56], interview is categorized into structured, semi-structured, and unstructured depending upon the way the questions are posed during interview. Structured interviews is one in which interviewer elicit information based on same set of question to each interviewee in the same way. A questionnaire can be used as an instrument to aid the interview process. This type of interview is suitable in situation where one require limited range of responses from interviewees [54]. Semi-structure interview involves open-ended questions asked sequentially by the interviewer to cover research topic [54]. This type of interview does not limit the interviewee to a pre-defined set of question as in case of structured interview. Thus providing freedom of asking questions to the participants involve in the interviews which results in getting broader picture of overall topic. A questionnaire is also used in this technique to give prior knowledge to interviewee about the research topic. This type of interview is suitable in situation where interviewer have little knowledge about what is happening in interviewee domain. A problem with this type is having risk of diverse information that is difficult to transcribe and is time taking. It is better to consider information which is related to research topic and interview goal. The proposed study will use semi-structure interviews for data collection process. It will help us in getting our required research questions in detail from interviewee's experience.

4.2.4 Interview Instruments

To support the semi-structure interview, a questionnaire was developed and communicated with the interviewees in advance before the meeting. The questionnaire was developed in such a way that it fulfills the interview goals described above based on research questions. In order to develop questionnaires, a literature survey was carried out in advance as a pre-requisite for the interview study. The literature survey cover requirements level defects, survey of various defect taxonomy for defect classification and DPTs used specifically for requirements level are discussed in detail before conducting interviews. Questionnaire can be found in Appendix A. The data and time of interviews meeting are decided in advance with interviewees. Interviews are conducted through face-to-face meeting and through Skype messenger. Time for each interview is allocated to 45-60 minutes approximately.

4.2.5 Interview Recording

To record interview data, tape recorder will be used. It is an efficient way of recording the interview complete information in original form thus avoiding loss of information. It is also useful for interviewer to focus more on eliciting information during interview instead of making a note of every interviewee's response. Besides this, we will collect key points during interview on notebook to remember what the interviewee has said during interview. It will enable authors to ask new questions based on what the interview described. This will also helps us to make a quick analysis of data to make sure that nothing is left out.

4.2.6 Interview Execution

In order to conduct interviews, various software organizations were contacted in Pakistan and Sweden, and three companies from each country were selected for data collection. The interviewees were selected based on criteria described above under subject selection. The date and time for meeting is decided in advance. A short description of research topic and interview questions is communicated with interviewees through email before the interview execution. This will provides each interviewee an idea about interview purpose and research topic. The interview will starts with formal introduction of interviewee (s), their domain, and their requirement engineering process and later interview questions will be asked sequentially. The time for each interview will be 45 – 60 minutes approximately.

4.2.7 Data Analysis & Validation

At the end of each interview, the collected data from tape recorder will be transcribed into MS word document. Data will be analyzed simultaneously using word documents and notes taken during interview. Both authors will analyze data using brainstorming and discussion and map the keyword in interview to relate research questions related to interview study. The process of analysis of data one after another give better control to authors over interview process. It is because of deficiencies from one interview help to improve and overcome them in later interviews.

In order to get the right data for each interview according to questionnaires the authors will do a short analysis during each interview. The authors will make important note of interviewee's response to make sure important things are covered. They will also make a quick review of interview questions at the end of interview to make sure that interviewee (s) answers all the questions and nothing is left out. Ambiguities after data analysis that lead to confusion about some information are eliminated through emails. This way helps the authors to verify the answer from the voice of interviewee. Besides that a copy of analyzed form of data will be sent to the interviewee (s) for validation purpose so that they can map and confirm it from their perspective what they described during interview.

4.2.8 Exact Interviews Execution

Before going into the details of empirical study results, it is important to have a look at how interviews were conducted exactly. The authors have given a step by step execution of qualitative interview design that is given below.

The exact execution of empirical study was based on qualitative interview design described in section 4.1. Based on planned study design, the execution of design was accomplished in the following steps

1) The interview study requires a strong background and related information of those RQ that could be achieved through interviews. The background part was achieved through the study of following topics in detail
    - RE process and related information (section 2)
    - Defect types from literature along with their definition (section 5.1)

- Commonly used DPTs and DP methods at requirement level (section 5.3)
- Requirements defect taxonomies and their purpose (section 5.2)
- Study of qualitative research design and interview techniques (Creswell [16])

2) The authors planned and designed the qualitative interview described in section 4. This section contains the complete detail of design
3) Authors searched for software companies and contacted the most relevant interviewees in these organizations. The selection of interviewees was based on criteria described by the authors in section 4.2.
4) Authors discussed and developed interview questionnaire based on RQs. The questionnaire was also communicated with supervisors for refinement and was updated accordingly.
5) Authors fixed data and time for meeting and a short description of thesis was also communicated with the interviewees to give interviewees an understanding of thesis background.
6) During interview execution, data was recorded with the permission of interview and some information was also collected on notes. As authors have proposed to use open-ended questions approach, so some run time questions were also asked and recorded when necessary.
7) After every interview, data was transcribed into MS word without any delay to avoid the risk of tacit information. The shortcoming in one interview was discussed among the authors and they tried to overcome it in the next interview
8) After converting all the data into MS word, data was analyzed against RQ.
9) The analysis process involved brainstorming and discussion on study results by the authors in the following steps (based on QDA model, see figure 6)
   - First of all things were noticed in the form of interesting terms or points and then these interesting things were given names. For example missing requirements defects, ambiguous requirements, major defects, minor defects, high or low rate of occurrence, training for RE process, dedicated RE department, informal RE process, SRS, and so on.
   - After noticing and naming the things the Collection and sorting processes were started in which the authors logically coupled related information into categories. For example missing requirements, ambiguous requirements or things like that were cohesively coupled in a category called “requirements defect types”, major defects or minor defects were grouped into “severity level” category, low and high rate of occurrence was grouped into “rate of defect occurrence” category, terms like training for RE process, dedicated RE process, and informal RE process were categorized into “RE process” category.
   - After collecting and sorting things, the authors focus on three important goals like 1) develop some kinds of sense from each group of data, 2) look for some special patterns, within the collections or across the collections (like similarities, differences), and 3) make some discoveries about a fact for which you are researching. Based on about goals the authors first thought about each category and tried to develop a sense and tried to find any possible relationship with other collections. Since data was collected from industrial interviews and reviewing the literature, so there was good opportunity to make relationships within the interview results, within the literature review results and across each other. On the bases of collected things the authors conducted comparison within Pakistani companies, within Swedish companies, between Pakistani and Swedish companies, within literature study, between industrial and academic study, and tried to compare findings with state of the art research.
   - The last steps involved Presenting information in a useful way (in the form of text, graphs and tables) that satisfy related RQs

4.3 Qualitative Data Analysis

The building of qualitative data analysis (QDA) is standing on three pillars that are noticing, collecting, and thinking. John V. Seidel has developed a model (see figure 5) that shows possible interactions among these three notes. He also called QDA as symphony of noticing, collecting, and thinking. He got this idea of explaining things in very simple way from his professor Ray Cuzzort who called statistics as symphony of two notes: mean and standard deviations. [83]

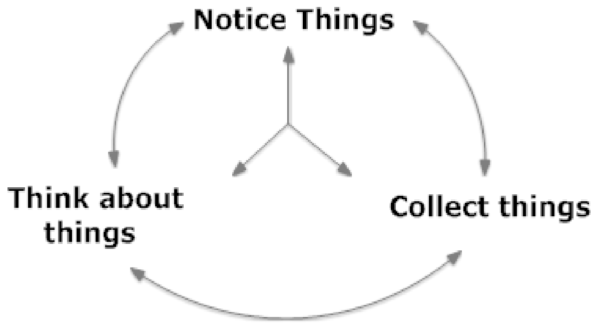


Figure 5: Steps in QDA [83]

The data obtained from qualitative interviews and from reviewing of literature will be analyzed on the bases of QDA model given by in figure 5. The figure 5 shows that QDA model is non linear and has following characteristics [83]

1). It is iterative and progressive because it is in cyclic form that can repeat the steps.
2). It is recursive because if you are collecting things then you might have needed to notice any other part to collect new things.
3). It is holographic because each step contains the entire process in itself.

Now we will put in plain words the meanings of noticing, collecting, and thinking in detail.

4.3.1 Noticing and Coding

Noticing is very similar when you are reading a book and highlighting interesting and important words, terms, or interesting things. It has two levels 1) Noticing on general level, and 2) Noticing the produced record. The noticing on general level involves observing the things, writing field notes, interview recording, and so on. In this way analyst just makes records of interesting things. On the other hand the second level noticing means to have focus on records to get information of interest. After going through the recorded data the analyst names the interesting things that he has noticed. This process is called coding things [83].

4.3.2 Collecting and Sorting

After noticing and naming the interesting things, the things need to be collected and sorted in the form of groups. It means the most relevant things will be categorize in the same category. For example, in jigsaw puzzle where we start by sorting the pieces of puzzle to make anything like a house, tree, or sky. [83]

4.3.3 Thinking

Next step is thinking about the collected and sorted things. It involves the observations of categorized things with different perspectives. At this point analyst keep in mind some goals like 1) he tries to develop some kinds of sense from each group of data, 2) he looks for some special patterns, within the collections or across the collections (like similarities, differences), and 3) he makes some discoveries about a fact for which he is researching.[83]

The figure 6 shows the whole qualitative study design that consists of industry interviews and reviewing of literature. It also covers solutions and study validation process. Before going to read about study results and other rest of material, figure 6 provides a roadmap to the readers.

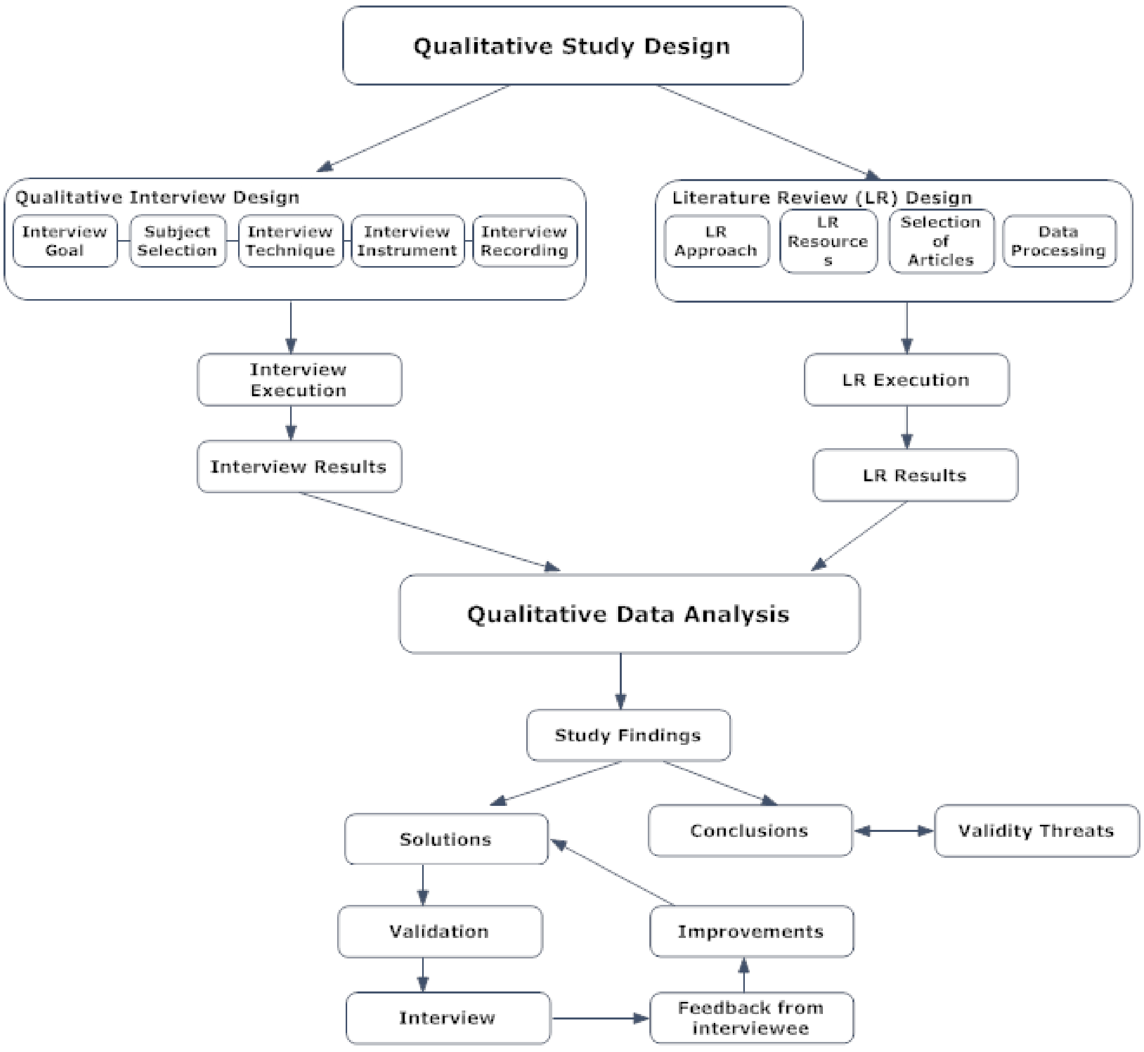


Figure 6: Study design

4.4 Validity Threats

Validation in research provides a mechanism to researcher to make an assessment of study finding whether they are accurate, reliable, and can be generalized to a population of interest in the specified research area [82]. Qualitative research validation deals with the extent to which the study results or findings are accurate and credible according to the participant involved in research [16]. To achieve this validity of study there are four different way to access the credibility of qualitative study described by [80]. They are credibility, transferability, conformability, and dependability. The current study is assessed according to these assessment criteria is given below

4.4.1 Credibility

According to Trochim et. al. [80], credibility involves the extent to which the results of qualitative research are credible and believable according to participant's perspective involve in the research. The purpose of credibility is to make sure that the study results are reasonable and realistic. To conduct rightful research study, the authors have designed qualitative research from two different but related areas of exploration. One aims at theoretical study by investigating requirements level defect based on research from academia while other focus on practical aspect in the form of industry interviews as a mean of data collection for the proposed study. Using both mechanisms of data collection give better control over research topic to attain research objectives. To make research credible,

- The authors used some reliable and quality database resources like IEEE, Springer link, ACM Digital Library, Science Direct, Engineering Valley, and Wiley Inter Science.
- To get realist and feasible results they focused on industrial case studies and experiments performed in industry related to our area of interest (DP in RE).
- They tried their best to get the most relevant interviewee and to ask the most relevant questions during interviews that cover each research question (see apendix)

Further a literature review was carried out by the authors in the following three areas;

- Research in literature about requirements level defects (see section 5.1)
- Taxonomies in literature specific to RE phase for the classification these defects (see section 5.2)
- Defect identification and prevention techniques to deal with these defects and their strength and weaknesses (section 5.3)

The literature review part not only helps authors to get broader insight into requirements levels defects but also provides a prerequisite for qualitative interview study. The interview part is designed and followed as described in section 4 and 6 respectively.

Authors have tried their best to perform a though literature review and conducted effective interviews with six companies both from Pakistan and Sweden with different domains for example company A provides content management and portal solutions, company C provides digital signage solutions, company D deals in mobile and communication devices, and so on. Most of the interviewees were well experienced and were appointed on relevant position related to RE. For example interviewee of company A was appointed as product manager having three scrum teams under him, interviewee form company B has four year experience in software development and involved in RE in many different projects like Air Traffic Control System (ATCS), and image tools, interviewee form company E was serving as project manager who has ten years of experience in software development, and so on.

On the other hand authors spent their best effort to conduct literature review regarding DPTs and DP methods. For example they extracted data from around 35 research articles that were related to DPTs and DP methods only as mentioned in section 5.3.

Date and time for meeting is fixed in advance with each interviewee. The research study proposed open-ended interview strategy. To aid this strategy, a questionnaire was developed (see appendix A) and communicated with interviewees before the meeting along with the background of thesis, so that they have complete idea about the research topic and information domain. Each interview data is recorded and validated by short review at the end of each interview to confirm nothing is left our and a short summary is also presented to them to confirm the elicited data. The transcribed document is also communicated back to the interviewees who validated the recorded data before starting analysis.
The authors have experienced some threats to credibility of the study that are given in following paragraphs.

Since authors are students and are not mature enough in creating a relevant questionnaire for interview study. For example they can miss some important questions, can make non realistic questions, and can misunderstand the meanings of interviewee's answer and so on. The

authors tried to remove this threat by consulting interview questionnaire with their thesis supervisor.
Although the authors tried to conduct interview in six companies from different domains but still they missed a lot of domains like e commerce, computer networks, embedded systems, telecommunication, artificial intelligence, biometrics, and so on. So study findings will be limited to the domains that the authors have already covered.

It is fact that some people have knowledge but they are not capable of explaining it in a good way. If an interviewee has this fault then it would be very difficult for the authors to extract right information from the interviewee. This threat was minimized by asking some runtime questions, and sometimes interviewee was asked to explain his point of view with real life example and scenarios.

Some interviewees can skip important information just because of limited time. In this way the information could be infeasible and incomplete. This threat was minimized by asking to the point questions.

The interviewee can also hide some important information just because of company's privacy policy. Before developing interview questionnaire the authors kept in mind that companies would not share their secret information in the form of documents. So questionnaire was free from the questions that can violate their internal policies.

There could be vast aggregation of relevant martial in different sources (like IEEE, ACM Digital Library, Springer links) and author can miss it just because of limited access rights, and limited time. The authors had enough time and had easy access to all sources (mentioned in section 4.1.2).

4.4.2 Transferability

Transferability deals with generalizing the research study results to some other settings, contexts or population [16]. As the proposed study is focused on academia as well as industry which provides useful information to learn about the type of requirements level defects. The study resulted in finding major defects that cause major rework if they are not identified and fix at requirements level. If these defects are identified at later stages of SDLC then they can cause major rework in term of more effort and time which obviously increase project cost. So the findings are supportive and useful for software industry to learn about the defects at requirements level so that special attention could be given to them to avoid their consequences. A part from these defects, there must be some actions in the form of techniques or methods to deal with them.

One of the major parts of study is the discussion about the defect identification and prevention techniques used to deal with requirements level defects. Although industry is using some of them but most of the organizations are not familiar with them and their importance. Authors have given a thorough description of DPTs, DP methods, defect identification techniques, and SRS reading techniques (in chapter 5) so that companies get familiar with them.

The study also provides support to organizations to learn about these techniques to improve their processes and quality of their products. The use of taxonomy helps organizations to learn about its structure and classification of defects in order to make certain decision based on defect classification. For example, classification of defect through taxonomy can help the analyst to take care of problems arising during E and A&N phases. It also provides help the tester to create test cases based on major defect types in the taxonomy.

One possible threat can be related to interviewee knowledge about DPTs and DP methods. It might be possible that particular interviewee doesn't know about DPTs, so we cannot generalize the things only on behave of single employee and there might be dozens or hundreds of employees with specific level of expertise and domain knowledge. For example when authors asked a question during interview with employee of company D in Sweden that "can we generalize things that you have told us"? They replied that you never do that because they have thousands of employees all over the world with different domains and departments. So the way of working might be different in different cities and countries. In response of this threat the authors have tried to minimize it by selecting the most experience and most relevant interviewee based on criteria described in section 4.2.2.

Another threat is that Pakistan has different ethnical groups of people than Sweden. Companies in Pakistan can have different working environment, trends, culture, and external environment. So study findings and solutions recommended by the authors (based on Swedish companies) might not work exactly in Pakistan as in Swedish companies and vice versa.

4.4.3 Conformability

The influence of researcher's judgment is minimized. The data and their interpretation are not figments of the researcher's imagination. [80]
As the proposed study consists of literature review and interviews, the authors use realistic approach to collect, interpret, analyze and document results (see section 5 and 6). The literature part focused on finding requirements level defects based on research from academia. The data is collected from literature using authentic database sources like IEEE and ACM etc (see section 5.1). The authors biasing and judgments are kept minimize by following the instruction and research design set initially in the research.

During qualitative interview part, the focus was on getting valuable data from experts in software industry. The authors use abbreviation instead of using organizations name (see section 6). This information can be audited and published with the permission of organizations. The interviews are recorded on tape and any external source apart from researcher can audit it.

On the basis of study results analysis, the authors proposed some solution (list of recommendation, defect taxonomy and modified N-Fold inspection technique). These solutions are validated by interviewees from company C and D respectively in section 7.5.

Since authors don't have industrial experience and don't know about environment of industry. It is possible that the authors create some non realistic interview questions due to lacking of software industry experience. The authors tried to remove this threat by consulting interview questionnaire with their thesis supervisor.

4.4.4 Dependability

The dependability assessment criteria refer to the ever-changing context or circumstance that are fundamental to the research study [80] [81]. This demonstrates that focus and strategies may change as the research proceeds [81].

During qualitative interviews, one of the major threat is the unawareness of software engineering terminologies for the interviewees e.g. interviewees from company D. Although organizations are using and following techniques for defects identification but they are not familiar with technical terms. Authors have designed the interview study and they followed it but the information provided by interview is abstract, less and sometimes irrelevant to research topic. Our focus remains same but we sometimes come up with scenarios to get into the information we are looking from interviewees perspective otherwise it might affect our findings.

It was also experienced that some of the interviewees reschedule the meeting (interviewees from company A & F) although it was already decided but due to interviewee personal problem we have to postpone it. This thing affect our schedule but to a smaller scale. Further the authors planed to give around two to three weeks to interviews but it took more than five weeks because interviewees had very tight schedules

4.5 Validation Design of Study Finding

The study results proposed during data analysis (in section 7) are credible according to author's perspective. In order to get opinion about the usefulness of results, authors have proposed interview with expert from industry to validate the study findings in order to determine their significance. The purpose of this validation is to ensure the credibility, effectiveness, and usefulness of study results so that they can be generalized in some context or population.

4.5.1 Validation Design

To validate the study findings, authors will conduct interviews with experts from industry who will help them in determining the significance of their study results. For this purpose, two interviewee from company C and D will be selected. The selection is based on criteria described in section 4.2.2. The data and time for interview will be fixed in advance. The time for each interview is expected to be 30 to 45 minutes approximately. Authors will conduct fact-to-face meeting with interviewees. To record the interview, tape recorder will be used. A questionnaire will be used for evaluation purpose. This questionnaire will be based on study findings and it will be developed by mapping questions (in questionnaires) to the proposed research finding explicitly. Before starting questions with interviewee(s), a short presentation will be given to interviewee about the

1) Background of the study
2) Reported requirements defects based on research from academia that originate from E and A&N phases of RE (see table 4)
3) Reported requirements defects investigated as a result of qualitative interviews (see table 6)
4) Use of defect taxonomy for classification (see figure 13)
5) Proposed list of recommendation given as a results of research study
6) Proposed techniques helps to minimize requirements level defects

5 LITERATURE STUDY RESULTS

5.1 Most Common Requirements Defect Types Reported by Literature based on E and A&N phases of RE

To identify most common defects based on research from academia, following research is carried out related to requirements level defects.

Jane Huffman conducted a study to deal with requirements level defects in [24] and it was based on fault-based analysis. He proposed a methodology for NASA base on requirements-based fault analysis by developing a requirements fault taxonomy and processes for tailoring that taxonomy to a specific project [24]. This methodology was used during verification and validation process. Historical data from previous projects was used to identify the most common types of faults, and risk analysis was performed to identify their likelihood consequences. According to [24], fault-based analysis can be performed early in SDLC prior to implementation. The primary purpose of this analysis is to identify the defects originating from the RE phase so that they can be prevented as early as possible to avoid rework in later stages. They identified 13 defects types related to requirements and grouped the related defects with defect types in a generic taxonomy [24].

The defect types described by [24] are incomplete decomposition, omitted requirements, improper translation, operational environment incompatibility, incomplete requirement description, infeasible requirements, conflicting requirements, incorrect assignment of resources, conflicting inter-system specification, incorrect or missing external constants, incorrect or missing description of initial system state, over-specification of requirements and incorrect input or output descriptions. The description of defects associated with each defect type is given in [24]. By applying the taxonomies for three different classes of project, it was found that for class B, 83% of defects were due to three types of defects which were incompleteness, ambiguous, or inconsistent. For class C, 83% of the defects were due to three types of defects incompleteness, omitted or incorrect requirements. Using same taxonomy for International Space Station (ISS) project by examining the defect data, it was found that the first three categories of taxonomy which include incompleteness (20.9%), omitted/missing (23.9%), and incorrect (23.9%) accounted for almost 80% of requirements defects [24].

Kosman has developed a two step methodology to get rid of two main defect types originating from requirements. This methodology can be applied on any kind of software system (like business, mission critical) that has Geographical User Interface (GUI) and needs requirements specification. They have categorized the requirements defect types into two main categories 1) Specification Generation Defects and 2) unwanted/unnecessary or incorrect user functionality and omitted user requirements written in specification and implemented in GUI. Specification generation defects type includes inconsistencies, ambiguities, typographical error, input/output mismatching, missing and incorrect data. [2]

Søren Lauesen and Otto Vinter have conducted a case study at Brüel and Kjaer requirements defects by studying a real time product. During case study they investigated a product after a few months of product release. They found around 800 defects reports and to avoid the burden of analysis they selected 200 defects (every fourth defect) for analysis. On the bases of interviews with developers and their own study, they found that 107 defects were from requirements and 93 were from implementation. These 107 defects were categorized into different types of defects. About 60 reported defects were due to tacit requirements (requirements that had not been written down, surprising requirements) that fall into omission defect type. About 20 defects belonged to ambiguous, wrong and forgotten requirements and remaining 20 reported defects were due to ambiguous requirements about external work packages. [25]

There are some common risks regarding software requirements. These risks might lead to a defected product. It is very essential to overcome these risks to reduce the probability of extra rework in the form of defect identification and fixing in later stages of SDLC. Thesis risks are 1) Insufficient User Involvement during elicitation and negotiation processes. In this risk developer thinks that he know what customer wants. On the other hand costumers don't want to spend more time for detail requirements. 2) Gold Plating, here developers can add a self-made functionality or feature to make customer happy or developers believe that customer will love it. So it can lead to unwanted or unnecessary requirements. It will waste resources during designing, implementation, and testing processes. 3) Creeping User Requirements risk happen when user frequently requests changes and developers don't response to that request 4) Ambiguous Requirements risk can cause bad deliverable, wastage of time when developer develops a wrong requirement and tester becomes confused about verifying the actual requirements 5) Minimal Specification risk takes place when marketing staff or managers create a limited requirements specification, perhaps a general concept of product instead of detailed requirements (might be omissions or missing requirements). They want developers to explore detail specifications when project is in progress. This risk put the developers in a tight position of frustration (they might be working under incorrect assumptions and with limited directions) and customers get disappoint to receive a wrong product 6) Overlooked User Classes takes place when requirements engineers don't care different classes of users. Some of them might use different subsets of features; have different frequencies of use, or different experience levels. It may lead to dissatisfaction of some users of the product 7) Inaccurate Planning risk occurs when inaccurate estimations are done like cost estimation. The major contributors of poor cost estimation are frequent requirements changes, missing requirements, insufficient communications with users, poor specification of requirements and insufficient requirements analysis. [4]

In another research article authors have shared their experience about aggregation of historical datasets that contained inspected defect data. Since aggregated datasets might be heterogeneous data that comes from different sources, at different periods of time, in different formats and have different terminologies, so to make it understandable and manageable they used different categorization schemes in aggregating historical inspection defect data. They collected defect data from 2529 inspections from 81 projects in NASA. For this purpose they visited five centers of NASA in USA. Each center was using different defect taxonomy and in some cases they were using multiple defect taxonomies. Their aim was to create a model based on historical datasets that describes the behavior of software inspection at NASA. They wanted to develop a unified defect taxonomy that can help in using historical defects to guide future development projects in NASA and this defect taxonomy has ability of backwards compatibility with existing data, captures domain specific elements of interests to NASA, holding desirable properties of good defect taxonomy. This taxonomy would facilitate the mapping of historical data to new sets of defect categories and would be easy to apply in future data collection effort. In this paper they created an initial draft of defect types for each work product like requirements, design, and code and test plan artifacts. [26]

They have categorized defects related to software requirements into seven types. These are clarity (requirements description may not be clear, what is required actually [7]), completeness (needed constraints or services are not missed [7]), compliance (problem with compliance to any relevant standard), consistency, correctness, testability (bad testable requirements mean unclear or missing requirements [7]) and others. Others mean any problem or defect that cannot be put into rest of defect types mentioned above that had logged during requirements inspections. [26]

A constant change in software requirements is one of the major causes of software defects and software industry is suffering from this issue. More changes in requirements make the requirements unstable. These changes are easy to manage in early stages like RE phase but in later stages of SDLC it has very bad impact. John's study about requirements changes has explained that a new feature in later stages of development life cycle has 50 percent more defects associated with it than those of the artifacts associated with original requirements [37]. We know that software development is dynamic process and this process is influenced by changes. It is noted that RE process even continued while software development is in progress and that leads to volatile requirements [27]. Requirements volatility is defined as tendency of requirements to change overtime. Volatile requirement is a factor that causes major difficulties in software development in the form of extra cost and effort (causes rework in later stages of SDLC), that's why this factor is called "the cost driver" [38]. Requirements volatility also has impact on defect density in code phase [28], project schedule and cost [29], quality of code, quality of project management and developer's capability [38]. In most of the companies project development is started with unclear, fuzzy and incomplete requirements. The major sources of change in requirements might be changing work environment, organizational complexity and conflicting requirements [27].

Lamsweerde in 2000 gathered empirical data by using survey from 350 US companies and over 8000 projects were investigated. This empirical study found that one third of the projects were never completed and one half succeeded partially. Partially means with partial functionalities, major cost overruns, and significant delays. When it was asked about the causes of such failures, the executive managers found poor requirements as the major source of problems (about half of the responses), the lack of user involvement (13%), requirements incompleteness (12%), changing requirements (11%), unrealistic customer's expectations (6%), and unclear objectives (5%). So changing requirements has overwhelming percentage (11%) of project significant delay, incomplete functionalities, and major cost overturns. [30]

In 1992 Michael Schneider, Johnny Martin and W.T. Tsai in [20] conducted an experimental study to verify the results from an earlier project which calculated fault detection rates in SRS. They appointed nine inspection teams to inspect the SRS by using N-fold inspection technique. All teams were motivated to find as many defects as possible for them. After experiment they found that rate of fault detection in requirements had been low by using formal inspections and advantages were achieved by using N-fold inspection technique that was created by the authors. The hypothesis of this technique is that "the N separate inspection teams do not significantly duplicate each other's efforts and that there is not a high degree of fault-detection overlap. Instead, each team neither finds a significant number of SRS faults nor located by other teams".

According to the hypothesis multiple reviews are needed instead of single review of SRS. If hypothesis is true then defect detection rate by using N-fold technique will be higher than single review inspection. The validation of this hypothesis was done by Martin and Tsai in a detail study (first experiment). They found that the average number of defects found by a single team were 25 out of 92 or 27%. When five parallel teams were appointed to inspect SRS, then fault detection rate raised up to 65% and after appointing ten parallel teams fault detection rate raised up to 80%. In this way defect detection rate was increased tremendously. But their first experimental study had major problem of uncontrolled experimental environment. Uncontrolled environment means differences in ability between teams, team's familiarity with inspection process, pace of inspection process and the number and types of defects assigned in their experimental study [19]. Therefore they decided to conduct second experimental study to avoid mistakes which were found in first experiment and to find reliable statistical data based on N-fold inspection. [20]

The SRS was similar in this experiment except with small change in control for the number and type of defects. During this experimental study they found different types of defects by using N-fold inspection in SRS. These requirements defects were categorized into two main classes that are called class 1 faults (Missing Information) and class 2 faults (Wrong Information Faults). Class 1 includes faults includes defect types such as 1) Missing Functionality or Missing Feature, 2) Missing Interface (how the system will interface and communicate with the objects outside the scope of the system that has been omitted), 3) Missing Performance (omitted system performance specification that is unacceptable for acceptance testing), and 4) Missing Environment (hardware/software/database environment that has been omitted). Class 2 faults include defect types such as 1) Ambiguous Information and 2) Inconsistent or Contradictory Information. [20]

Søren Lauesen, Jan Pries and Otto Vinter have conducted a case study at Brüel & Kjær to prevent requirements level defects. They studied defect reports of previous projects to understand issues in the RE process that can lead to defects. They identified three main classes of requirements defect tpyes which are missing requirements, volatile requirements and ambiguous requirements by developers and testers [31]. The defects are classified and analyzed by using Boris Beizer Taxonomy describes in [32]. The Beizer taxonomy is categorized into ten major classes each of which comprises of three levels [32]. We will discuss taxonomies later in detail but the category deal with requirements level defects in Beizer taxonomy is “Requirements and Feature” which further breakdown into main defects of requirements completeness, presentation, volatility and incorrect requirements each one is further divided into third levels to describe related defects under each category [32].

Defect taxonomy is an effective means of identifying and analyzing the type of defects in the development process. It provides basis for corrective actions both for present and future perspective of product in the form of prevention whose primary goal is to reduce the number of defects in the product. [33] According to a case study 51% of defects are related to requirements [31] out of which usability issues were dominating up to 68%. Other problems were understanding of third party software and their associated defects and functionality issues were in common. The defects from defect reports did not reflect all of defects found in Beizer Taxonomy, so it was modified little bit with new defects subcategories at lower level of Beizer Taxonomy [31]. Requirements related errors were then classified based on error source that cause defects and quality factor [31]. These quality factors were based on ISO 9126 quality factors model that consider the quality characteristics of functionality, usability, reliability, maintainability, and portability, each one of which are further divided into sub categories [31].

The purpose of considering the quality factor in classification is to get deeper understanding of quality requirements by mapping quality factors like usability and functionality issues into those stated in ISO 9126 because functionality and usability issues are dominating up to 68% as discussed above [31]. In most of the defect reports the major source of defects was the tacit requirements during elicitation phase that caused missing or misunderstood requirements [31]. To deal with the problem, they studied different 50 DPTs based on hit-rates and proposed to use Scenarios and Navigational Prototype Usability Test, Daily Tasks in the requirement elicitation and validation phase by training their employee with the proposed techniques. By applying the proposed techniques 27% of requirements related defects were eliminated, whereas according to quality factors, requirements issues have been reduced by 36% [31].

Another case study conducted in same organization (Brüel & Kjær) by Otto Vinter and Soren Lausen is similar with the previous one but the primary purpose was to find the most common types of defects and problem in DPTs related to requirements [34]. The study was based on analysis of defect report to identify most common defects types. The report was analyzed using interview with people involved in development. To make classification of defect they used Bezier taxonomy with little modification. According to authors, the reason for modification was lack of some categories in taxonomy that didn't fit well with some of today's programming language and tools as Bezier taxonomy was developed in mid 1980's [32][34].

During analysis they founded that most of the defects originated from the RE process. Majority of defects were due to missing, incomplete, tacit, assumed, and volatile requirements in the implementation phase. The primary defects were not concerned with SRS instead they were due to missing and tacit assumption about the requirements in the elicitation phase [34]. According to authors, about quarter of defects were in "requirements and features" category of Beizer Taxonomy [34]. They also classified the defects according to source of defects and perform a closer analysis of requirements related defects using quality factor also describes in [31] to identify quality issues like usability issues which was reported to be 64%. Other defects were deal with understanding and cooperating (28%) with third party software found during interface analysis, while other issues account for 13% [34]. To provide effective preventive actions, they performed cost/benefits analysis of potential prevention techniques and mainly focused on techniques that addressed the requirements elicitation issues [34]. They proposed to use scenarios and navigational prototype usability test and daily task after training their employees with these techniques [34] are also described in [31]. By applying proposed techniques, an overall improvement of 27% reduction in defect report was reported with 72% decrease in usability issues [34].

In another research article authors described that inspection in requirements specification phases can figure out inconsistent and incorrect requirements before they go for design and implementation. Most of the requirements reading techniques (such as ad hoc, checklist, defect based reading) do not pay attention on particular aspects of the SRS and put all requirements information on the same level of importance. In this way the identification of all types of defects in the entire document becomes foggy. Researchers at the Experimental Software Engineering Group at the University of Maryland have created Perspective Based Reading (PBR) that provides a set of software reading techniques and can find defects in SRS written in English language. PBR believes that the importance of information in requirements might be different for different uses of SRS. [22]

There are three major uses of SRS in SDLC such as 1) A description of the customer's needs, 2) A bases for the system design and 3) A point of comparison for the system test. These uses provide foundation of perspectives for reviewing SRS. For example designer needs correct and clear requirements with details for the major component of the system that is under review. The tester focuses on requirements testability and wants appropriate information to build a good test plan. The customer or user of the system uses SRS to know the correct and complete accomplishment of all system functionalities. It was found that PBR can find following defect types in SRS 1) Missing Information, 2) Ambiguous Information, 3) Inconsistent, 4) Incorrect Fact (a requirement fact that cannot be true in particular condition of the system, wrong behavior), 5) Extraneous Information (unnecessary or unused information), 6) Miscellaneous defects (errors like including requirements in wrong sections). [22]

5.2 Requirements Based Defect Taxonomy

Defect taxonomy is an approach to learn about the types of defect occurs during SDLC [68]. If there are number of defects reported during software development or during operation than it is difficult to relate which one belong to which phase in SDLC and activity in that phase. Taxonomy provides a simple and efficient approach about the types of defects [68]. It is an effective approach of learning between projects through defect elimination of same types and helps to avoid them in future. Taxonomy in general is the "classification of things into ordered groups or categories that indicate natural, hierarchical relationships among categories" [69]. This word is a combination of two Greek words "Taxis" meaning arrangement of things and "onoma" meaning name used for representation [69].

Defect taxonomy maintains systematic arrangement of related defects identified by unique names in order to represent and retrieve information about them. Different taxonomies have been developed in different fields of sciences depending upon their requirements and information to organize i.e. business taxonomies, medical sciences, software fault taxonomies, and software security taxonomies are common examples [70]. Taxonomy in software aims to categories the defects among SDLC based on core phases i.e. requirements, design, implementation, testing etc. representing the group of defects in each phase which further describes the defect types within each group and so on until the defects lie under a certain defect type. It helps to understand the defects mechanism and provide an ease to put a defect under a certain type belong to that defect. For example, defects related to requirements are put under requirements which are further classified into types e.g. incompleteness, consistency, ambiguity etc. These types can be further narrow down into defects associated to each type e.g. incomplete requirements, incomplete attributes, or incomplete feature or interfaces etc. The level of classification can vary until it satisfy the complete defects under a certain type but the relationship between defect and types must be consistent and coherent [32]. It should be simple and easy to understand so that it can be easy to make decision about a particular defect position in the taxonomy [32]. Some common defect taxonomies related to requirements phase of development life cycles are discussed below;

5.2.1 Orthogonal Defect Classification (ODC)

A part from defect taxonomy for classification and analysis of defect data, IMB in mid 90's, proposed Orthogonal Defect Classification technique to categorize defects and their possible causes in software [71]. Using ODC, defects are classified based on defect attributes consisting of defect types, defect trigger, source and their impact [72]. This classification helps in analyzing the data by understanding the categorization of defects and possible causes thus help in making action plans for focused defect. It also improve process activities by understanding of defects distribution over phases through mapping of defect types to the development activities (i.e design, coding, etc) [71][72]. The overall purpose of using ODC is to find the deficiencies in process by mapping the defect types back to the development activities to know their root causes. So that deficiencies can be eliminated that causes the defect to surface. ODC mainly focuses on design and implementation defects with limited and fixed number of defect types and triggers that is difficult to use for requirements [21] [71] [73].

5.2.2 Boris Beizer Taxonomy

It defines a four level classification of software defects using fine-grained framework used for categorization of defects [32]. Major categories related to requirements in Beizer taxonomy are shown in table 2; [32] [74]

Table 2: Requirements defect types in Beizer Bugs Taxonomy

1xxx Requirements
11xx Requirements Incorrect
12xx Requirements Logic
13xx Requirements, Completeness
14xx Verifiability
15xx Presentation , Documentation
16xx Requirements Changes

It is a comprehensive and life cycle oriented bug taxonomy covering requirements in detail along with other phases of SDLC. Thus it provides complete information of defects for the whole development process. This taxonomy classifies bugs in more detail with four digit number to represent the level of detail for bugs from major categories to defect types to their respective defect. A four digit characters i.e. xxxx is used as mask for each category as shown in table 2 which are further assigned number as proceed from top level to downward. The first level describes major nine categories covering development activities consisting of requirements, feature, implementation, testing etc. Each category is associated with defect types specific to that category that is mutually exclusive, non-conflicting and consistent to avoid the chance of putting the defects in wrong category. From table 2, it is clear that there are six defect types associated with requirements that were proposed by Beizer in 1990. These types are further associated with defects that best represent the type. The requirements classification through Beizer taxonomy provides useful information about the types of defects that can be easy to use and implement for requirements level defects. This way of identifying the nature of problem helps to know the actual reasons behind the problem [32] [74].

5.2.3 IEEE Taxonomy for Software Anomaly

IEEE uses a generic term “anomaly” that is used for all terms i.e. error, faults, failure, problems, bugs, defects, flaw, incident or glitch to communicate all these types. According to IEEE, anomaly is “any condition that departs from the expected that comes from documentation or from someone’s perception or experiences” [75]. It is also generic and life cycle oriented taxonomy of software bugs. The classification process proposed by IEEE consists of four sequential steps which include [75];

1) Recognition
2) Investigation
3) Action
4) Disposition

Each step undergoes through administrative activities of recording, classifying and identifying the impact. The purpose of these administrative activities is to record the anomalies related information in each classification process. This information provides useful help to make improvement, corrective and preventive actions. Whether it belongs to a process or work product. For example, the classification process starts with recognition of anomalies. In this process environment in which anomaly has occurred and related information is record. Further important attributes of anomalies are classified through a classification scheme. Finally, information about the consequences of anomaly shall be record by the person who recognizes the anomaly. This information about anomalies continues until the last steps of classification process. The classification scheme divided the anomaly among project phase, activities in each phase and the suspected causes that lead to that anomaly. [75]

5.2.4 HP-Defect Classification Scheme (DCS)

In 1986, Robert Grandy and Deborah of Hewlett-Packard (HP) proposed a simple defect classification scheme (DCS) for defect categorization [10]. In this DCS, information about defects is established using two forms; one for the classification of defects in term of defect severity, method of discovering and symptoms while the other form establishes the resolution of detected defects based on information gather in classification form. These both forms are maintained under defect tracking system use to track and resolve the defects in the system. Defect tracking system is the part of HP metrics program that is used to determine the causes of defects in order to improve the development process. The resolution form set priority of each defects along with proposed solution that focus on how to fix the defect and the person who fix it. They proposed that team should have a clear understanding of why the development team needs to track defect data that affects the software. [76]

The classification scheme of HP focus on three dimension 1) Origin (where) of defects in the phases of development process i.e. requirements, design, code, etc, 2) Type (what) of defects in specific phase e.g. specification and functionality defects related to defects, hardware interface, software interface under design phase, etc. 3) Mode (why) of defects bases on defect type that associate the defects under a defect type e.g. missing, or unclear mode may relate to defect type software interface that was either missed or unclear [76]. HP classification scheme in general is

Origin (Where) -> Defect Type (What) -> Mode (Why)

The problem with this scheme is support of limited number of defect types in each development with limited number of defects associated with defect types. Defect mode in HP-DCS considers missing, unclear, wrong, changed and better way to fit them for all defects types encountered in each development phase thus making it generic independent of phase.

5.2.5 Requirements Fault Taxonomy

NASA in 2003 proposed a requirements fault taxonomy specific to RE for its projects based on historical data for understanding the most common and repeatable defects along with their root causes [73]. The taxonomy helps to identify most common faults that would be overwhelming. This taxonomy is used a part of requirements based fault analysis for early development phases i.e. requirements prior to design and implementation. It is based on classification of faults in requirements by considering the characteristics and relationships between faults. The analysis is performed as a part of verification and validation process. The taxonomy is based on two basic principles 1) fault categories are mutually exclusive, 2) fault categories are not specific to a particular language, environment, or system development approach [73]. Thus it is generic in nature and can be applied to any software system. According to [73], the requirements faults originate in requirements phase i.e. elicitation, analysis and found in the SRS. The major categories of requirements faults are described in table 3. The subcategories associated with those described in table 3 can be found in detail in [73].

Table 3: Requirements defect categories in requirements fault taxonomy [73]

1. Requirements Faults
1.1 Incomplete decomposition
1.2 Omitted requirements
1.3 Improper translation
1.4 Operational environment incompatibility
1.5 Incomplete requirements description
1.6 Infeasible requirements
1.7 Conflicting requirements
1.8 Incorrect assignment of resources
1.9 Conflicting inter-system specification
1.10 Incorrect or missing external constants
1.11 Incorrect or missing description of initial system state
1.12 Over-specification of requirements
1.13 Incorrect input or output descriptions

5.3 Requirements Defect Identification & Prevention Techniques and Their Weaknesses

There are lots of defect identification and prevention techniques and methods those help in stopping requirements defects to penetrate them into later stages of SDLC. If requirements defects would mistakenly allowed reaching the customer or tester then these defects would be very costly to fix in the form of tremendous rework [4]. Authors have conducted both empirical and literature study for defect identification and prevention techniques that have been using during requirements analysis phases. There are also some methods those help in preventing requirements defects. These methods try to prevent defects in different phases of the RE processes like E and A&N. Authors have given details of these techniques and methods with their advantages and disadvantages. In this way readers would have good idea about attempts those were made to prevent defects in current industry. During literature study authors tried their best to keep focus on industrial case studies and research papers those have industrial experiments

5.3.1 Defect Identification Techniques

5.3.1.1 Prototyping

In the RE process, SRS is used for requirements validation in the presence of customers or users to make sure that the requirements are realized in the right way. The problem with this approach is that users find it very difficult (by just reading specification document) to visualize how the system will perform required tasks or functionalities. It becomes more difficult when developing system is supposed to be used by large numbers of users having conflicting requirements and users with weak background of system knowledge. They feel uncertainty in the fulfillment of requirements that they have put forward. It is also very difficult to know whether an SRS is complete, consistent and ambiguous and if these threats prolong to downstream stages of SLDC then it will be more costly to correct them [42]. To overcome these problems a lot of methods and tools have been working. Tools are divided into two categories 1) system specification languages and 2) graphical tools. Formal methods are also another approach which uses formal languages like problem statement language (PLS) and requirements specification language (RSL) to develop SRS. Similarly graphical approaches (such as information flow diagram, and ER diagram) has been using to develop quality SRS. The use of formal specification languages (FSL) has been described in detail in section 5.3.2.1.

The above methods and tools for requirements specification leave a gap between users and developers of a system. Prototyping can remove this communication gape because user can give vital feedback to the developer on the appropriateness of SRS. Even the circulation of SRS among different stakeholders in the organization often gets no useful feedback and it is very difficult for users to have a thorough reading of such a lengthy document. That’s why they find it boring and consequently results become misty. The cost of accommodating changes increases tremendously as the software development moves into later stages of SDLC [42]. Prototyping can minimize this cost by getting user feedback early in RE phase. Prototyping also helps in developing applications that are more users oriented. Prototyping approach is also useful for training users about how to use the system but it is necessary to keep the prototype up to date. This advantages also is compared with cost factor, it will not be used if cost of maintaining the prototype is exceeding planned budget. [42]

Prototyping is implemented after the development of preliminary version of the SRS. In this way developer has good opportunity to take vital feedback from users to improve the quality of SRS. The concept behind the prototype development is to show the users backend functionalities of the system that are not visible for them. If developer will follow this concept then they would be able to get beneficial feedback from the users. Prototyping should be stopped when time and cost required making modifications (in response of user feedback) to the prototype have been outweighed. On the other hand if developers and users feel its usefulness they can continue it again. In short prototyping approach ensures that SRS is complete and correct by only spending less than 10% of total cost for the project. For ambiguity and consistency checking, RSL can be used. [42]

5.3.1.2 N-fold Inspection

The SRS is very important RE work product because it defines the needs and boundaries of the software product. It also acts as an agreement between customers and developers. It plays pivotal role in the success or failure of the developing product [19]. It is bases for rest of project planning, design, and coding and is foundation for system testing and user documentation [4]. Finding and fixing software problems after product release is 100 times more expensive than finding and fixing during RE practices or during design phase [19, 42]. The investigation of SRS is very important to avoid rework in later stages of SDLC because problems in products (like SRS) of early phase have very bad impact on product development. Defect detection in SRS could be done by different methods.

Formal inspection is very effective and efficient approach and particularly in case of requirements inspection phase it can find inconsistent or incorrect requirements [22]. N-fold inspection based on formal inspection (Fagan Inspection) described in [39] it replicates inspection activities by using N-fold inspection. It is used to identify faults or defects in SRS as early as possible just to avoid them to be occurred in later stages of SDLC. In N-fold the same artifact (SRS) is given to all N independent inspection teams. A single moderator handles the results of all inspection teams. In the end of inspection process moderator gathers effort of all teams and records detected faults in a database. Since there is probability of a fault or defect to be identified by multiple teams, so the moderator writes each fault once in the database. The logic behind N-fold inspection is that a fault that is not supposed to be found by single team can be found if multiple teams are working on single artifact [19]. It is also hoped that different inspection teams will find different defect or faults, in this way large number of teams will find more defects [39].

It is very important to choose the best values of N in N-fold inspection. Sometimes by increasing inspection teams the rate of fault detection does not increase with respect to added inspection team members [19]. The best value of N based on three factors 1) availability of teams, 2) cost of additional teams, and 3) potential cost for not finding a defect during inspection. Study of Kantorowitz et al., (1997) have proved that N-fold inspection does not depend on type and size of the system to be developed. They have suggested that more teams for inspection and more inspection experts in each team can assure better performance [39].
Advantages

- The overlapping of fault detection by different team is minimal
- Early detection of faults in SRS becomes more effective
- Two or more teams inspecting same document can identify more faults as compared to single team.

Disadvantages

- Some faults were not found during inspection of SRS even by multiple teams, it means inspection process is good but not good enough.
- Some faults of SRS cannot be discovered during inspection that needs execution in the form of design or implementation. These faults can be identified by using formal specification and design.
- A flawed inspection team cannot perform well in the identification of faults
- Multiple teams in N-fold inspection is costly but it provide substantial benefits

For requirements analysis, inspection has been proved an excellent technique [22]. During inspection process, inspectors need some sort of reading techniques for SRS. These reading techniques might base on experience and knowledge, business objectives, product goals, defect taxonomies, particular scenarios, specific perspectives and so on.

5.3.1.3 Ad Hoc Reading

This technique is used to take a general viewpoint of reviewers. It does not offer any support to the reviewers. They use their own knowledge and experience to find defect in a document. It does not enforce a specific process to perform inspection [39]. Participants are guided during inspection sessions to find defects in SRS. All participants follow defect taxonomy. For example they can use following defect taxonomy.

- Omission Defects: this category includes missing functionality, missing environment, missing performance and missing interface

5.3.1.4 Checklist Based Reading

It seems more systematic than Ad hoc reading. In this reading technique participants are given a list of questions that are answered by the reviewers or they tick predefine important issues that need to be checked. This list of questions helps the inspector (reviewer) to carry out good inspection process. For each project there should be unique checklists. For each individual type of document to be reviewed and for each individual type of product, there must be specific type of checklist. [39] For example if currently a company is developing real time application that put reliability, availability and performance attributes on high priority then checklist to review artifacts should focus these attributes. In the same company if another project say Management Information System (MIS) is going on then checklist to review artifact will cover issues related MIS.

5.3.1.5 Scenario Based Reading (Defect Based Reading)

Defect based reading (DBR) technique was developed to find defects from SRS. In this technique, inspection is done based on different scenarios that are defined according to defect taxonomy in use. Defects are categorized into different classes and for each class of defect a set of questions are developed. Scenarios are also considered to focus a specific view point that helps to indentify particular type of defects. [39] For example a company is developing Railway Track Crossing system. During the inspection of SRS for this project, reviewer can make a scenario like “when train reaches 5Km away from the railway track barrier then barrier will start to come down. Let if barrier stops in the middle and do not come down completely then what could be problems”

Porter and Votta in 1994 and Porter in 1995 have conducted multiple experiments in which they compared scenario based reading (SBR) with checklist and ad hoc reading techniques based on defect identification rate. The findings of experiments proved that scenario based or DBR has higher rate of defect identification than checklist and ad hoc based reading approach and checklist reading was no more effective than ad hoc. Practically it was proved that SBR captures 35% more defects than checklist and ad hoc approaches. [48]

In 1995, Gough et al has performed a large scale study in market industrial environment in which he used scenarios based reading technique in Fagan’s inspection. Each inspection team was consisted of five reviewers having particular role. Gough found that reviewers captured 2-4 defects per hour. Author claimed that SBR is good mean of requirements elicitation. Fusaro et al in 1997 also conducted and experiment and compared SBR with checklist and ad hoc. His results seem not in favor of SBR technique. He pointed out that average defect identification rate for SBR was not significantly different as compare to checklist and ad hoc reading techniques. [39]

5.3.1.6 Perspective Based Reading

It is an enhanced form of DBR or SBR technique. Instead of using defect classification (as in case of SBR) PBR focuses the stakeholders point of view and their needs regarding system. Scenarios are developed on the bases of stakeholder’s point of view [39]. It provides a set of procedures to reviewers (developers) that can help them to know the problems of requirements inspection. The theme of this reading technique is that the requirements have different information which is more or less significant for different uses of the SRS. It is very important to find true users of inspect-able artifact (like SRS) and to know how they will use the artifact but this selection of users is different for different project or organizations. There are some examples of different perspectives to use SRS in later stages of SDLC such as 1) Description of the customer's needs, 2) A basis for the system design, and 3) A point of comparison for system test. PBR guides the reviewers during requirements inspection to answers the two important questions. These questions are 1) what information in these requirements they must be checked? And 2) how can they find defects in that information? [22].

Basili et al. have performed some experiments to know the effectiveness of PBR on SRS in NASA. He found that there is no major difference of reviewer's who were using PBR and those who were using other reading techniques like checklist. But PBR reviewers performed well on generic documents. Laitenberger and DeBaud have also performed deep experiments to find effectiveness of PBR. They also found no significant difference in performance of reviewers when they were using PBR on code document. On the other hand Shull et al. said that PBR is better for reviewers who have some experience. According to above discussed of authors, it is proved that PBR reviewers can capture more defects as compared to the reviewer who use less systematic and less structured approach for review. They further added that PBR is more systematic, focused, goal oriented and tailor-able [39, 48].

5.3.1.7 Usage Based Reading (UBR)

Effectiveness and efficiency of fault detection can be improved by using reading techniques such as PBR, DBR, CBR and ad hoc reading techniques. User perspective has got values in software development by different methodologies like use cases in object oriented development and operational profile testing. User perspective can be used during requirements inspection when inspector prepare for inspection meeting. Inspectors can use user oriented approach in reading SRS just to confirm the quality of specifications from user's point of view. There are two important attributes of UBR, such as 1) use cases, and 2) prioritization. Use cases helps the reviewers to inspect SRS and prioritization helps in sorting out most important functionalities from user's perspective. During inspection the reviewers get a prioritized set of use cases and inspect the software artifact. Similarly a fault will be critical if user will consider it critical. [49]

In an experiment that was conducted by a group of students [49], they proved that UBR technique is more efficient in detecting faults than CBR and UBR is also efficient in finding different defects or faults than CBR (that is popular in industry). There are also some drawbacks in UBR those are given below.

- It is not feasible for real time system because real time system are explained by event based tables or diagrams
- It does not cover business goals
- It does not cover quality attributes of the software system

5.3.1.8 Function Point Reading

It is the enhanced form of SBR technique. To strengthen the inspection process for commercial systems, Benjamin Cheng and Ross Jeffery have developed scenarios based on Function Point Analysis (FPA) that is called Function Point Scenarios (FPS). Authors conducted an experiment in which they used SRS as inspection artifact instead of code and implementation documents. They proved that FPS technique is more efficient to inspect crucial areas of the software artifact (like SRS) than simple SBR technique. [50]

5.3.1.9 Metric Based Reading Technique

Besides the inspection of SRS, it is important to find defects in notations (use case diagrams) in which that SRS was written down. On the bases of this recommendation, Metric Based Reading (MBR) technique was come into being. Its main goal is to inspect the SRS and to identify specific types of defects in use cases. MBR is based on a set of heuristics that are used during requirements inspection process. These heuristics are developed based on some structural properties of use cases (that are easy to measure) that could be early indicators of some particular type of defects in use cases like incompleteness, ambiguity, misunderstanding, and lack of conciseness in use cases. There are lots of use case metrics such as Number of steps of the use case (NOS), Number of actor action steps of the use case (NOAS), Number of conditional steps of the use case (NOCS) and so on. These metrics are used as defect-proneness indicators. Each heuristic defines a threshold level for a use case metric, if a use case has values more than this threshold then the probability of the use case being defective increases. Authors had conducted an experiment to validate this technique. They found that it was more effective in the detection of defects than CBR technique. However it was not proved more efficient because it take to much time to be implemented properly. [51]

5.3.1.10 Inspection Using Error Abstraction

In [52] a group of authors had conducted an experiment to validate PBR technique. During experiment they found that it is very hard to classify, quantify and defining individual fault. Reading techniques that concentrate on individual fault have to face problem of fault classification, quantification and definition. There are some drawbacks of concentrating of faults during inspection process such as 1) same fault can be described in different ways, 2) some faults could be linked with totally different parts of the requirements, and 3) similar faults can be in single group of fault. If we take researcher's view point then they recommend concentrating on errors (mistake by human) instead of individual fault, so that reviewers can get rid of above mentioned problems [52].

There are some potential advantages for this approach such as 1) communication about the document can be improved by focusing on area of functionality that document author has specified incorrectly rather than focusing on every individual mistake, 2) if reviewers will focus on basic misconceptions then they can learn what is actual problem and how they can prevent it to happen again, and 3) since error are higher level of abstraction than faults and they could be made available in the organization to other reviewers and developers. In this way prevention from these mistakes can be avoided. Authors had proved by a controlled experiment that this process of error abstraction appeared feasible but the effectiveness of the process could not achieve author's expectations. [52]

5.3.1.11 Goal Oriented Requirements Analysis

Requirements analysis should not rely on only understanding and modeling the functions, data and interfaces for the new system but they should explore alternatives and evaluate them with respect to business objectives and transitively these objectives will meet business goals of the system [46]. Goal Oriented Requirements Analysis (GORA) technique provides a platform for traditional requirements analysis techniques (inspection, ad hoc based reading, prototyping etc) for their better performance. So that traditional requirements analysis techniques can identify and evaluate alternative ways of meeting business goals. It provides a way of refining organizational and technical objectives so that more and more alternatives (alternative solutions to meet an objective or goal) can be explored during requirements definition [46]. For example our generic goal is to achieve better security of ASS (Airport Security System). We can decompose these goals into multiple sub goals like achieving availability, reliability, performance, confidentiality and integrity of the system as alternative solutions. It is not necessary that all alternative solutions will be considered but we can select some of them by using OR and AND operators.

Each of these sub goals can be decomposed into further sub goals to identify more alternatives to refine organizational and technical objectives. In case of ASS we can identify more alternatives by decomposing its second order sub goals (availability, reliability, performance etc). We can decompose performance attribute into response time, information sharing, fault tolerance ability, computer memory in use, resources to be allocated in the execution of software and so on. Similarly we can decompose availability attribute into time to repair, time to rejoin the system, time to failure, duration of availability of the system and so on. In this way a hierarchy of goals is developed that help the requirements analyst to have simple form of analysis. There are five steps [46] in GORA technique.

1) Goal analysis, where authors decompose functional requirements into an AND/OR hierarchy as shown in figure 7 below. In this figure the decomposition of Schedule Meeting goal has been shown. There are a lot of alternatives and each alternative has specific plan to satisfy the goal. It is marked with single arc that shows AND operator and double arc shows OR operator. It means satisfaction of a goal will be accomplished with the satisfaction of all sub goals or by the satisfaction of any sub goal respectively.

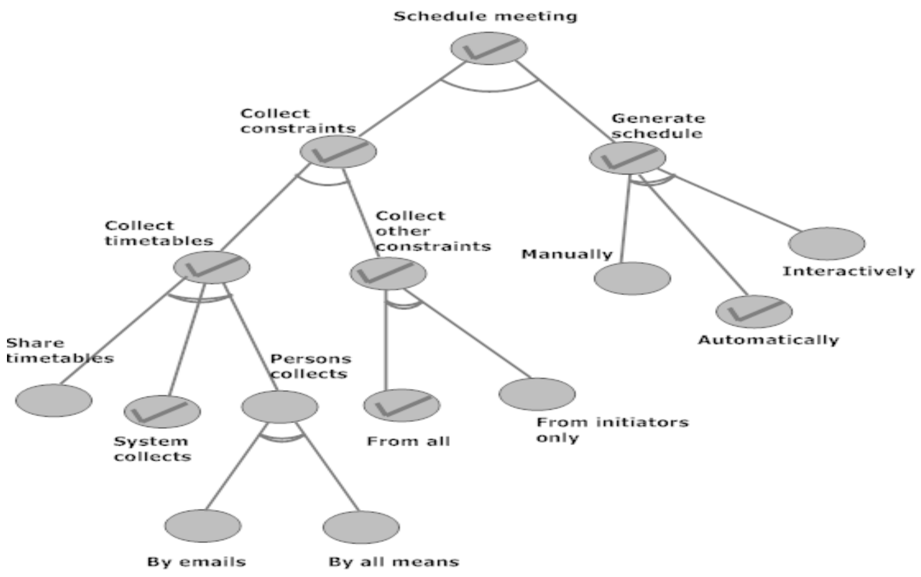


Figure 7: An AND/OR decomposition that depicts alternatives for achieving the meeting scheduling goal [46]

2) Soft-goal Analysis, here the quality attributes associated with a system are decomposed into soft-goal hierarchy. Figure 7 shows non functional requirement for a system that is "system should be highly usable". It means usability is the soft-goal and that can be decomposed into hierarchy of sub soft-goals that is shown in figure below. This hierarchy is composed of AND/OR operators and also positive operators that shows a loose relationship as compare to AND/OR. Positive operator indicates that a soft-goal is positively influenced by its sub soft-goals. It means a soft-goal will be satisfied if it will get more positive influence from its sub soft-goals. However in case of more negative influence it will not be satisfied by its sub soft-goals.

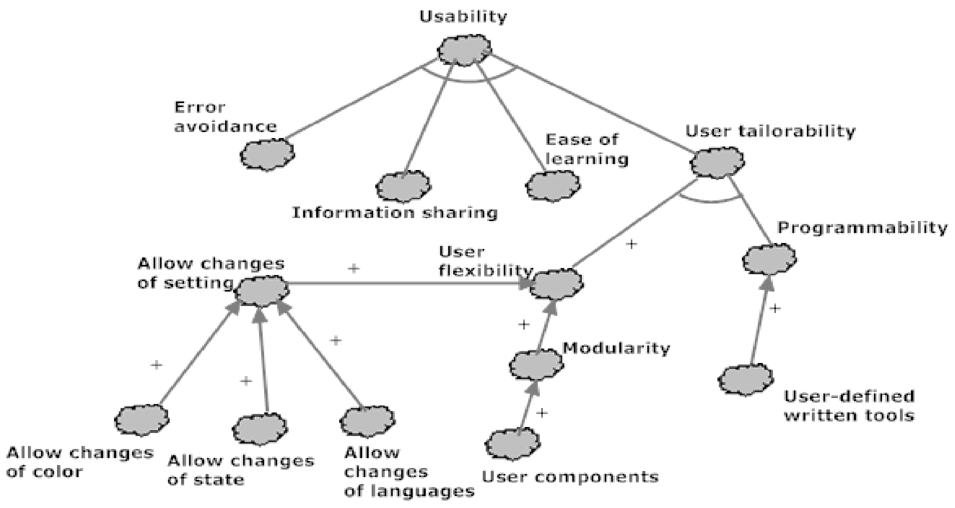


Figure 8: A partial softgoal hierarchy for usability [46]

3) Since quality goals have frequent contradictions between each other. If one goal is achieved then other might get less attention. For example security and user friendliness, and high quality and low cost. Soft-goal correlation analysis is performed to identify positive or negative relationships among conflicting soft-goals.
4) Goal correlation analysis, which identifies relationship between goals and soft-goals. So we have to combine figure 7 and 8 on the bases of best relationships between goals and soft-goals. Figure 9 shows a possible set of relationships for refined version of the Schedule Meeting goal and soft-goals such as Minimum Effort and Quality of Schedule.

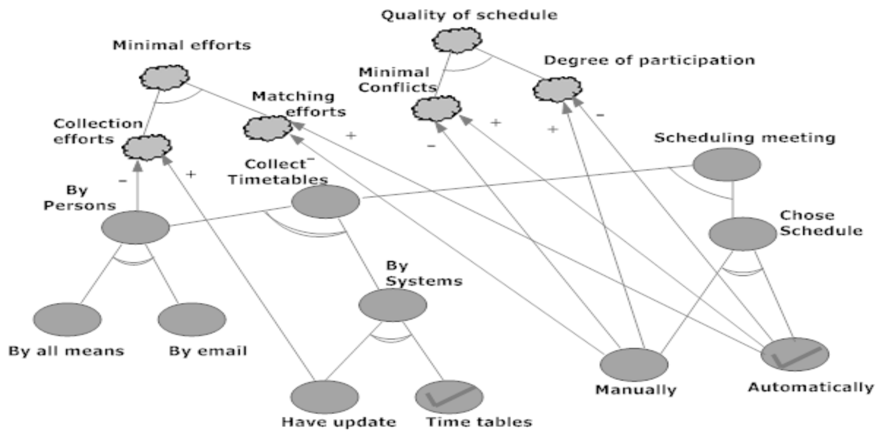


Figure 9: The result of goal correlation analysis for schedule meeting [46]

5) In the last step, evaluation of functional goals decomposition is performed in terms of soft-goal hierarchy that was constructed to meet quality of the system. This evaluation is performed by selecting a set of goals and soft-goals and that can satisfy all functional goals and expected quality of the system.

5.3.1.12 Attributed GORA Technique

There is an advanced version of GORA technique that is Attributed GORA (AGORA). In this technique system attributes are valued in digits (such as contribution values and preference matrices). Requirements analyst attaches these values with edges and nodes respectively in a goal graph. The contribution values of an edge show that to what extant a sub goal contributes to achieve its parent goal. On the other hand preference matrix of a goal represents the preference of a goal for each stakeholder involved. AGORA can help the requirements analyst to find inconsistency among the goals, to analyze the impact of requirements changes and to choose and adopt a goal from its alternatives. Requirements analyst can also judge the quality of SRS on the bases of its quality attributes such as correctness, unambiguousness, completeness and so on. [53]

5.3.2 Defect Prevention Methods

We have given detail of some methods that support DP in different phase of the RE process. We explained how these methods work and what kind of problems they have. These methods are given below.

5.3.2.1 Formal Specification Method

Formal specification method has been an important method in designing, validating, documenting, communicating, reengineering, and reusing solutions and formality helps in obtaining higher-quality SRS within such processes [41]. Formal methods are based on mathematical formulization. In the RE process formal methods are used in requirements specification phase. For informal or semi formal methods such as structured methods (like object oriented analysis and structured analysis) text, diagrams, tables and simple notations are used. On the other hand formal aspect demands mathematically formal syntax and semantics to specify system requirements in the form of system function and behavior. FSL are composed of three components that are syntax (Specific notation with which the specification is represented), semantic (how the language indicate system requirements or the real and correct meaning of requirements represented by a language) and relation (it defines the rules that show which objects properly satisfy the specification). Some examples of FSLs are 1) using model based notions such as Z and Vienna Development Method (VDM) and 2) based on process algebras such as Communicating Sequential Processes (CPS) and LOTOS [7]. It can be defined as “a formal specification is the expression, in some formal language and at some level of abstraction, of a collection of properties some system should satisfy.”[41] Formal specification removes ambiguity and encourages quality in early stages of SDLC. Because it depends on mathematical formulation so it is possible to verify the incompleteness, inconsistency and correctness checking formally specified requirements. The use of formal methods enhances system user’s confidence because by using formal methods requirements actually represent user desires. [7]. Formal methods are more useful where safety and security is critical. Unfortunately the formal methods could not get popularity among developers [41, 7]. There are some drawbacks such as

- Difficulty for system users and software developers in understanding the notations used in specification.
- Difficulty in specifying some aspects requirements for example requirements related to user interface.
- Software management feel hesitate in using new techniques that are not frequently practiced [7]
- Only functional requirements or properties of a system are formalized by using formal specification techniques (BNF-Style Specification, algebraic approach, and model based approach) and non functional properties are left behind the wall. It means formal specification techniques have limited scope
- There is no clear separation between essential properties of the system (functional requirements), assumptions about the environment of the system considered and properties of the application domain. All of them are mixed together in SRS.
- Formal specification techniques generally require extra ordinary expertise in formal systems and particularly mathematical logic. Due to this requirement the use of formal specification methods is limited in industry
- A typical user, who is going to approve SRS, will face problem in understanding the SRS that is written in a formal language [42].

5.3.2.2 Structural Analysis and Design Technique (SADT)

SADT is a diagrammatic notation that is designed to make the people understand a system or to describe the system. SADT provides rigorous expression of high level idea that previously had been too foggy to treat technically. It does not solve the problem but it allows people to understand, manipulate or check problem elements. SADT has arise from a premise that "The human mind can accommodate any amount of complexity as long as it is presented in easy-to-grasp chunks that together make the whole" SADT identifies these chunks, make the structure of these chunks visible and model them into a describable form . It has been playing good role in defining requirements, specifying functionalities of a system and in problem analysis. It has been applied to wide range of complex problems from real time communication to process control, system software, project management, software system specification, simulations etc [43, 44].

SADT consists of 1) the box-arrow diagramming language of structural analysis and 2) the design techniques. These both principal parts are strongly related to each other [44]. It provides two types of techniques 1) for performing system analysis, and 2) for performing system design. It also provides a process by which these techniques types are applied on requirements definition (requirements analysis) and system implementation. The most important aspect of SADT is Graphic Techniques. The SADT graphic language gives an idea about initial constructs from which analysts and designers of the system can get orderly structures of any required size. There are some simple notions by which graphical representation of a system or initial constructs of a system is possible, such as boxes and arrows. Arrows don't represent data flow as in case of dataflow diagram but it represents interfaces between parts (Boxes) in the same SADT model or in different SADT models. Whole represents a diagram in which multiple boxes (maximum six boxes or parts), natural language names, some other notations and arrows exist as shown in figure 10 below. Same graphical notations are used in both activities and data [43].

An SADT model depicts a well organized sequence of diagrams, each with concise supporting text. The high level diagram shows big picture of the subject and each low level diagram show a bit detail of well constrained topic or problem. This process continues until appropriate detail of the given topic (such as software specification) has been achieved as shown in figure 10. By this continuous process of problem decomposition SADT model represents a hierarchal structure of the system. However its depth is bounded based on its advantages and contents of SADT for a system are bounded by its viewpoint

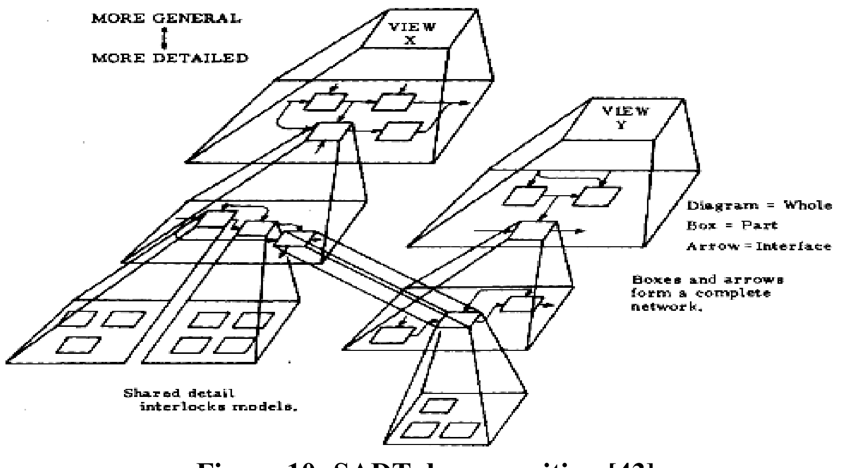


Figure 10: SADT decomposition [43]

Each low level diagram remains in connection with high level diagram logically. In this way the logical relationship between all higher and lower level diagrams of a system remains correct. Requirements definition (requirements analysis) needs dedicated team work. For this purpose SADT has mentioned its own way of establishing titles and assigning roles. For example in requirements analysis the author would be analysts, trained and would have experience in SADT. The interaction between assigned roles helps in meeting communication needs of requirements definition, for regular and critical reviews and for understandable and current documentation [43].

During the construction of SADT model for a system, if anyone wants to give suggestion or wants to make a change during review, he can write directly on the copy of draft. This change becomes the part of project files. When author and commenter reach a satisfactory position the work is reviewed by a committee of senior technical and management personnel. After this review documentation is produced as the model has evolved. The final SADT model makes the project highly visible. It is now easy for management to study the requirements in a top-down manner from basic overview to relevant levels of details [43].

SADT provides detailed graphical diagram of the system specification that is understandable for the system users so that they can give valuable feedback to the developers. From users point of view the best way of knowing whether his requirement have been met for a particular system, is hands-on use of the system [42].

5.3.2.3 Goal Based Requirements Analysis Method

Normally, requirements analysis explores relevant data and functions that a system will contain. The features and functional data of the system might be represented in the form of entity relationship diagrams, data flow diagrams or one can use object oriented approach. Goal Requirements Analysis Based Method (GBRAM) provides a vision to cover maximum aspects of the developing system. It provides a platform for traditional requirements analysis to perform well [46]. GBRAM enforce to categorize, decompose, and structure goals as requirements. It is very often in the organization that their goals remains unclear and are not provided easily. So to identify organization goals it is very significant to have as good information as possible to understand the domain, organization, process, and system [45].

Next question is where can we identify goals during the RE process? What entity or process will be responsible to achieve that goal and what constraints are there for each goal? To find answers of these questions we have to elaborate different artifacts (SRS, design description etc) and aspects of the RE process. In [45] they tried to give answers of above questions. They have explained that goals can be found by searching for statements which seems to guide design description at different levels with the system of organization, by searching for action and operation words from customer or any stakeholder's description document. For example for a Library Management System customer can describe actions and operations words like "reserve, delete, search and update" and we can extract goals such as reserve book, delete old book, and search a book and so on.

Goals responsible and stakeholder must be identified as early as possible. For example responsibility of goal *search book* is the responsibility of students and librarian. Constraints are also important because they provide some useful information for achieving a goal and keep the achievement of goal in definite scope. They can be identified by looking for dependencies and by searching for connectives *like before, during and after.* [45] For example for goal "search book" constrain might be as "student must have student ID for searching book".

Goals evolve with the passage of time, because the mind of stakeholder can be changed, or their goal priorities can be changed. There are two types of changes 1) goal elaboration and 2) goal refinement. Goal elaboration can be handled by some techniques such as identifying goal obstacles, analyzing scenarios and constraints, and operationalizing goals. When change happens in goal priorities then scenarios helps in the evaluation of new goal prioritization. On the other hand goal refinement is needed when contradiction in goals occurs, when goals are merged into a sub goal categorization, when constraints are identified, and when goals are operational zed. These issues can be removed by enabling different methods and techniques such as determination of pre and post condition of a goal, elimination goal dependencies and parallelism, goal categorization and usage of goal elaboration. [45]

Now we will discuss problems and issues that were found during implementation of GBRAM on Career Track Training System. It was observed during implementation that GBRAM is extremely fragmented, tremendously effort and time consuming, and extra human resources utilization [45]. Authors have conducted interviews from six software development companies and authors found that no one is using GBRAM. It means it is not widely used method.

On the bases of GBRAM in [46] a group of researchers have developed GORA technique. They enforce to explore alternatives ways to satisfy system requirements that had been explained in section 5.3.1.11

5.3.2.4 Object Oriented Requirements Analysis

Object oriented approach for system analysis has been popular for few decades. There are some modeling techniques such as data flow diagrams, RE diagrams and state transition diagrams that have been using widely in software development process. These techniques give proper shape to system specifications that object oriented system analysis becomes possible. In this way system analyst becomes able to capture the rich information that need to be model, analyzed, and understood before putting software system into implementation phase. Early requirements analysis techniques were made on the bases of structural programming concepts. As the structural programming has been converting to object orientation, so requirements analysis techniques have got influence form this conversion. Object oriented Analysis (OOA) techniques models the real-world environment, that means an environment comprising of people, work processes, material things, and software systems [47]. For example if Library Management System (LMS) system requires OOA, identification of classes (student, librarian, books etc), attributes (for example for class Book, name of the book, author name, ISB no. etc) related to each class, services (for class librarian, services might be search book, reserve book, delete book, register a student and so on) that a class can provide, and relationship between classes (such as librarian can reserve a book for student, book can be returned by a student) is very significant. After it the idea of whole system is modeled by using modeling techniques such as data flow diagram, ER diagram, and transition diagrams.

Object Oriented Requirements Analysis (OORA) is getting popularity because it provides better understandability of requirements specification and supports in object oriented design and implementation [67]. During OORA the above discussed diagrams and concepts with natural language helps different stakeholders (in case of LMS stakeholders would be students, requirements analyst, librarian, university, developers, project manager and so on) to be agree on the relevant objects and relationships among them. How can we model objects to accomplish requirements analysis? We can find the answer from [67] where an object model is given that tells how requirements can be made analyzable. The main points of this model are 1) identify the object classes from the requirements statements, 2) identify association between classes, 3) identify object attributes, and 4) organize classes using inheritance to share common structures. These steps can be put into an iterative loop just to get complete and correct object model. After it you can apply bottom up or top down or both approaches to build a hierarchical object model.

It is possible that late in design phase, the requirements object oriented model might get a change in the form of enhancement [47]. For example if a LMS needs to have track of information about very old books just to vanish them from the shelf, then there will be addition of a new class such as oldbooksinfo. There are some drawback of this approach that are given below

- Empirical study has proved that GORA technique can provide more detailed requirements definition than OORA techniques.
- It emphasis on static modeling and for real time system (such as distributed system) only OORA is no enough [67].
- It does not handle non functional requirements, that way in [46] the author did move from OORA to GORA because GORA handles non functional requirements in the form of soft-goals.
- It seems that designing takes more attention than real problem

5.3.2.5 Joint Application Design (JAD)

JAD also known as Joint Application Development was developed in 1970s by IBM [35]. It is considered to be an effective approach for DP in early phases of development. The basic idea of JAD is to conduct a joint session of system clients, users of different background and people within organization who manage and develop the system requirements. JAD sessions bring information sources that interact with the system on one platform to share information, discuss business needs, propose opinions, discuss problems and policies, and user's expectation toward the proposed system [35]. These sessions will be ended up with agreed and joint decisions about system constraints and refined requirements.

To improve the quality of requirements, software engineers use structure analysis techniques i.e. data flow diagrams, entity relationship diagrams and data models etc to describe the stakeholders' requirements into logical system representation that is understandable to them and developer as well [35]. Prior training and preparation are provided to the stakeholders in this regard which make them able to understand these technical terms [35]. This process resulted in the form of a collective and collaborative SRS with combined authorship [34]. Thus we can say that it gives a clear picture of system being developed to all those who will use this documents during this process and later in development process. It can be concluded that this team base approach of exploring system requirements in a collective and cooperative environment can helps;

- To eliminate communication barriers by providing an environment where everyone who interact with the system can participate to express his needs.
- Resolve conflicts of varying stakeholder's needs toward systems in an open atmosphere by providing them opportunity to discuss and negotiate.
- Brings developers, users and system analyst on a single point in the form of agreed set of requirements in order to eliminate ambiguities which further improve later phases due to clear understanding of system requirements.
- Helps in eliminating potential defects related to requirements in the form of correctness, ambiguity, missing and incomplete requirements through continuous customer's involvement. [35]
- Give management an opportunity to control project scope, budget and schedule by clearly understanding the customer needs and organization's limitation.
- Good replace of interviews i.e. instead of having a series of interviews to know stakeholders needs, make a joint session of them to know system requirements. [35]

As JAD is a team based approach that establish sessions in the form of meetings, it requires prior homework (in the form of training to employees and users, and access to right people for sessions especially from client sides), scheduling JAD session, pre-defined roles of people and their responsibilities (facilitators, requirements analyst, developers and users), meeting agendas, managing JAD sessions and people commitment to make JAD session effective [35] [57]. The JAD team can improve efficiency of the RE process by making reasonable decisions, explore business need together, develop business model, analyze business need, negotiate each other to reach consensus (in case of disagreement), and document agreed set of requirements [35].

The philosophy of JAD is to refine the RE process together with the stakeholder's so that there is less chance of requirements issues in later stages of development. Getting right people from stakeholders (end-users, executives, and developers) and their involvement is also complicated in order to get the right job. There is also need of training and effective leadership for successful JAD sessions. [57]

5.3.2.6 Cleanroom Methodology

Cleanroom methodology (CRM) was built on the idea of defect free development that came from electrical industry. During manufacturing of silicon chips in semiconductor industry, they proposed to use defect free environment in the form of clean room which is free of dust particles, humidity or chemical vapor or any other particles that affect the production of semiconductor [58]. Using same idea of electrical industry for software development, form the basis of cleanroom. Instead of dealing with defects after deployment or during testing, it is better to eliminated them during manufacturing or development process i.e. zero defect during development.

To achieve this, cleanroom provides a complete and structured set of engineering activities similar to typical life cycle activities (planning, designing, implementing and testing etc) but with focus and use of mathematical approach [58]. It is iterative in nature and build product in small increments [58] [59]. The principle that uses cleanroom for defect free software during development is based on mathematical techniques used during requirements specification, verification and design. We will stick around requirements and design phase in cleanroom to see how cleanroom is an affective methodology for DP in early stages of life cycle.

At RE level, it uses formal method for formal specification of requirements instead of using natural language which is often misunderstood and misinterpreted [58] [59]. Formal method is based on mathematical and logical formalization that is used to specify requirements in mathematical form. Formal specification is describes in detailed in section 5.3.2.1 [58]. Along with mathematically established formal specification at requirements level, it uses correctness verification techniques in a team review to make sure these requirements are correctly specified before the implementation of software [58].

In cleanroom, the specification team is responsible for analyzing and specifying stakeholder's requirements. They produce two different specification documents i.e. functional and usage specification. The function specification provides basis for each increment which carry out design and verification cycle using functional specification. Usage specification is based on usage scenarios for all possible system usage. This specification is used to generate test cases and provide basics for testing process. The function specification describes external system behavior and mapping of all possible users inputs to the expected output that describes system behavior in all circumstances. [58] [59]

Cleanroom process recommended a box structure approach for specification and design while functional verification is used to confirm that designed is correctly implemented with respect to specification [58]. The functionality of system is describes in term of objects using box structure through three level of abstraction which are demonstrated by black view which describes behavior view (input, output, and behavior of object), state view describes the finite state machine view and clear view which describes the procedural view [58] [59] [60]. These three different views are implemented by using Design Box Language (BDL) used to define the functionality [60]. From the above discussion it is clear that though this technique is efficient due to use of mathematical notation for requirements representation but it require trained people that are familiar and expert in using it. Further, people indented to use SRS should also know about formal method in order to understand the requirements. It is also difficult to communicate the SRS with the customer. From [58] [59], it is found that cleanroom focuses on use of formal method during analysis so cleanroom provides aids to requirements analysis as compared to elicitation phase of RE.

5.3.2.7 Quality Function Deployment (QFD)

QFD is considered to be an effective DPT which focuses more in early phases of software development especially requirements and design. The main focus of QFD is to satisfy customer's needs. According to [61], QFD is used as a systematic process of identifying and prioritizing customer requirements and incorporate these requirements into subsequent process and product specification. QFD focus more on requirements and design phase of development life cycle by making a correlation between customers and design requirements by identifying primary functional needs [62]. This correlation is established through matrix based approach between customer's requirement and design requirements that drive those customer requirements [61] [62]. The QFD process is achieved through a planning tool called House of Quality (HOQ) [61] [62]. It is matrix based approach for specifying customer requirements and design parameters. The term HOQ is based on series of process where each process is considered as a room. Each room presents information in the form of matrix. Collectively, these rooms form the shape of a house. There are five rooms and a roof in HOQ as shown in figure 11. The activities and information contained in each room is given below also described in figure 11 [61] [62];

1) Customer's requirements – This room describes the customer's requirements written in their own words describing what they want called "voice of the customer". It deals with the requirements elicitation from customers. To enter into to HOQ these requirements must be structured. It contains the prioritized list of customer requirements. Requirements are prioritized based on customer's importance which assign weight to each requirements to demonstrate their importance.
2) Technical requirements – This room deals with requirements that describes design requirements that drive how these customer requirements will be accomplished. A QFD design team is responsible for specifying these requirements. These requirements are also specified in the same manner in matrix like customer's requirements. This matrix lies below the roof of the house.
3) Roof – The roof of HOQ is a matrix above technical requirements where they are cross checked to determine correlation between engineering activities.
4) Correlation between customer and design requirements – This matrix is used to measure the relationship of each customer requirements and design parameter to measure their significance. This matrix will demonstrate how a certain design parameter will be associated to a certain customer requirement.
5) Planning matrix -A planning matrix is used to demonstrate an overall weighing of product by comparing customer's priority of each product feature with the competitor's products. This planning mechanism helps management team to find the importance of customers' demands, competitor's strategy, market demands and improvement areas for the current product [61].
6) Target matrix - It is the final process of HOQ which is based on conclusion drawn from information from other rooms of HOQ. The QFD team discusses and makes collective decisions. It contains an assessment of technical and cost estimation of technical requirements that are further used for resource allocation based on these estimations [61].

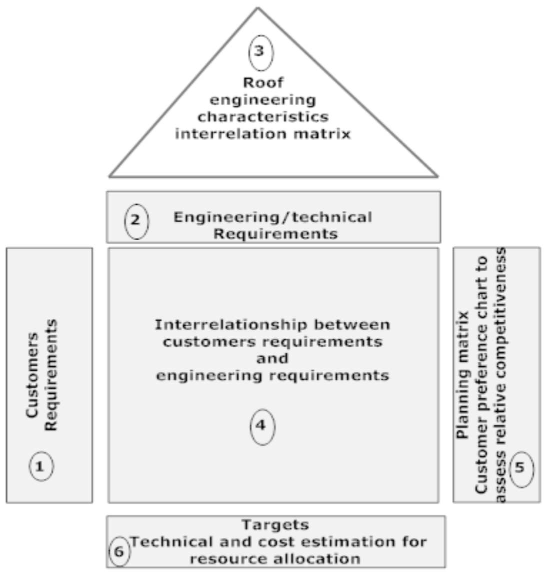

3
Roof engineering characteristics interrelation matrix
2
Engineering/technical Requirements
Customers Requirements
1
Interrelationship between customers requirements and engineering requirements
4
Planning matrix
Customer preference chart to assess relative competitiveness
5
Targets
Technical and cost estimation for resource allocation
6

Figure 11: House of quality in quality function deployment [61]

QFD focus more on customer satisfaction throughout the project. It is a complex method and develop matrix for everything from customer requirements to engineering specification. For each matrix that measures the importance or correlation between two entities in HOQ, it uses different calculation, scale, symbols and statistical method to value that object. We can say that HOQ is a tool used in QFD to determine customer and engineering specification during product development. It is a team based approach with greater involvement of analysis and design team. It also requires a lot of work in the form of competitor’s product for planning and performance of product. Moreover, the size of matrix is directly related to customer requirements. The greater the number of requirements for a system the larger will be the size of HOQ, which requires larger among of effort. Besides its complexity, the accuracy of requirements and design parameters that are used to convert those requirements into development is very efficient. As it compare customer’s requirements through their voice to design parameter that play an important role in their implementation. Further, considering competitors perspective, customer importance, design parameters, organization existing products helps in improving product design and performance. Comparing design parameters with product feature helps in establishing a consistency between requirements, design and implementation. One of the major benefits of QFD is to overcome requirements and design problems i.e. inconsistencies, ambiguity, clarity, incompleteness early in life cycle through matrix-based approach [62].

5.3.2.8 Participatory Design

Participatory Design (PD) as name describes is a collaborative and collective approach that focus on end-users involvement in development process. The primary goal of PD is customer satisfaction which is achieved through collaboration of end-users who are considered as domain expert along with designers and developers [63]. The participation in PD means discussion about the system and end-users requirements through active user participation in the requirements elicitation process where they collectively share information and identify issues. PD focuses on actual users of the system and the environment in which they work [64].

PD, the "Scandinavian approach" to system development, focuses on much stronger involvement of the end-users than JAD does, facilitating a mutual learning process between users and designers, and joint experiences into a simulated work situation [65]. During requirements elicitation, requirements can also be gathered through PD workshops. The focus of PD is to consider the end-users viewpoint about the product thus mainly focus on customer satisfaction. This could be achieved by gathering and involving them in a workshop to get their needs and demand about the proposed system. This will give more and more information from end-users perspectives. A workshop in PD is an efficient way of getting important customers requirements. In PD workshop, developers, designer, and customers work together to identify system requirements and design a solution that realize customer's needs [64]. This participation with end-users involvement can lead to efficient and usable system design.

According to [66], the people intended to use the system can play a vital role in designing the system because they have better understand of system i.e. they are domain expert. Through collaborative participation, changes can be minimized as users are directly involved in designing system. Thus, the rate of volatility will be low if requirements are initially developed clearly with the collaboration of customers. User's participation in design can improve prior knowledge about system before development of actual system, unable development team to develop realistic system according to expectation and reduce resistance to changes [63]. Both JAD and PD are similar and are design approaches with user involvement during software development [35] [57] [63] [66]. According to [57], the different between JAD and PD is the participation criteria in term of participation selection, involvement, technical staff and facilitators participation, structure and development speed along with goals setting for collaborative system design.

The goal of JAD is to produce a high quality design while PD focuses on social context of system domain with respect to users who indented to use it [35] [57]. PD emerges as result of a workplace democratic moment in Scandinavia in 1980 to improve the social environment of worker in order to enable them to efficiently use the system. This gives the idea of worker participation to improve the quality of workplace because of workers have better understand of domain [57]. So both JAD & PD focus on user's participation in the workshops. They both facilitate low-documentation and visualization method like prototyping, models for data representation during workshops [35] [57].

6 EMPIRICAL STUDY RESULTS

6.1 Most Common Requirements Defect Types Reported by Industry

6.1.1 Company A

Company A is privately owned Swedish software Company. It was founded in 1994 and is leading company in Content Management and portal solutions through the platform EPiserver. The company is a Microsoft Gold Certified Partner and holds AAA-ranking by Dun & Bradstreet since 2000. Company A has its main development centre in Stockholm and they have their development teams in Norway, Denmark and United Kingdom as well.

6.1.1.1 Interviewee

The contacted persons for company A is the second product manager working on company's major product and directly deals with requirements. Product manager is responsible for the RE process and overall product development. There are three teams working under the supervision of product manager and each team has five members. Product owner is the main source of providing requirements to product manager for a certain release. Product manager along with other participant selected from different team based on their experience establish the RE process right from requirements acquisition to final development of SRS which provide basic for later life cycle activities. The participants involved in the RE process include product owner, product manager (participate as analyst), tester, and developers. The interviewee is directly involved in the RE process and according to him the company wants improvement in the RE process. Being involved in the RE process, he knows issues related to requirements before and after the product is released to the customers.

6.1.1.2 Defect Types and Their Reasons based on E and A&N

It was quite different from what we were expecting from interviewee. We thought that interviewee will answer our questions very simply, if we will ask about defects originating from requirements then he would answer simply but case was quite different. Whenever we asked question he answered describing real life scenarios that he had experienced in his company. Interviewee has little bit different experience about the severity of requirements level defects with respect to definition of major and minor defects types as described in table 6 in section 7.1. He said that requirements defect types like missing, ambiguous, inconsistency, and extraneous information are minor defect types. He did consider omission defect type as major. Since they freeze the requirements before implementation and after requirements validation which means volatility factor is not serious here and change requests are put into next releases. However they think that requirements are not perfect and there might be some missing or unclear requirements during implementation phase. If they find an unclear requirement during testing phase then they reactivate the RE process. So they expect that the RE process can be activated again. They believe in dealing with defects as early as possible. They use scrum methodology, defects found in one sprint are fixed in that sprint before proceeding to the next sprint. They mostly identify defect during implementation phase. The organization uses their own definition to describe the severity of customer reported defects i.e. broken functionality (medium priority defects), critical functionality (highest priority defects) and entire functionality (low priority defects). Out of these types, critical and broken functionality defects are serious and communicated with developers who fix them. There is no mechanism of training of their employees regarding RE process and they rely on their experience and knowledge. To avoid repeatable defects happen a results of human mistakes, they use guidelines to make sure that same mistakes will never happen again. The interviewee was curious to hear about DPTs like cleanroom, JAD and QFD but eager to learn about them.

6.1.1.3 Techniques Reported by Company A

Company A do not have a dedicated RE department rather they have a support team responsible for communication with customer. RE process is based on experience and knowledge of people responsible for it. Organization has iterative process for requirements validation and they create wire frames (basic user interface mock-up) for each feature or functionality on a paper or whiteboard or by using prototyping tool to verify that they have draw right requirements or not. They prevent a defect by just having analysis on it and provide some guidelines about it. They formulize the problem by using different methods like pair programming, and code review. For document review they use checklist for requirements analysis. They compare features and functionalities against the checklist. To make the requirements feasible they use two techniques such as 1) Decomposition of requirements, and 2) prioritization. To write requirements they use dialogs. For requirements analysis they spend time in short meetings and discuss the problems. They don't know about DPTs that are mentioned in section 5.3.

6.1.2 Company B

Company B is Pakistan based IT company with eight year of experience in off shore and custom development in major software technologies. The organization provides outsourcing services and works with different projects. It deals with different projects and has its client in Pakistan and in Germany.

6.1.2.1 Interviewee

The contacted person from company B has a four year experience in software development and involved in RE in many different projects which included Air Traffic Control System (ATCS), image tools, downloader FTP/HTTP and data base updater application. The interviewee was involved in RE activities in some project as a requirement analyst, developer for developing prototyping for requirements validation as well as a tester. The interviewee was aware of RE issues and it was interesting and beneficial to get our research questions from his experience both as a developer and requirement analyst. The interviewee was a valuable resource for us due to his variant role in different project gives us a broader picture of defects from different perspective in the form of requirements analyst (who is aware of defects at requirements level), developer and tester (defects during testing related to requirements.)

6.1.2.2 Defect Types and Their Reasons based on E and A&N

The interviewee identified unclear requirements as a major defect because requirements are not initially clear which causes a lots of problems. The reason for this problem is lack of investor involvement, limited time allocated for RE and for validation purposes and political factors within organization. It was also experienced that most of the defects were reported by end-use in the form of missing features and these features are hard to fix if they are related to functional requirements. They can cause malfunctioning and even system crash. Due to missing requirements at initial phase of life cycle they are hard to identified but uncover when customers report them due to some problem with software. This means that they were neither implemented nor tested because they were not specified in the SRS. Sometimes these missing features are reported by QA team. Inconsistencies and conflicting requirements are resolved at requirements level and there is a very little chance of their appearance and consequence in later stages. Infeasible requirements are settled down by negotiation with customers. In case of a change in functional requirements, it has direct influence on system design. The reasons for this change are lack of understanding, and the initial work is not done in the right direction. Defects are mostly reported by customers (e.g. missing requirements) and according to interviewee sometime they are greater than 60% but if these defects are tackle at requirements level their rework will be 5%.

Defects reported during requirements validation are more than 74%. The interviewee believed that they believe in DP strategy which is ensured though a well defined RE process. According to interview, the defect identification rate from E and A&N phases of RE is approximately 60% to 70%.

6.1.2.3 Techniques Reported by Company B

The company B has a dedicated RE department and they involve customers during RE process. Three to four people are involved in RE process with 5 to 6 years of experience. Company B uses different techniques during the RE process. The requirements elicitation phase starts with studying the existing system. The requirements analyst is responsible for studying domain knowledge, conduct interviews with relevant people who interact with the system. People involved during the RE process are requirements analyst, client (system users), developers and management. During elicitation phase, the company uses more than one technique. The company uses interviews and brainstorming techniques during the requirements elicitation phase. After study existing system, the requirements analyst uses questionnaire during interview with system users. The questionnaire contains set of key questions related to new systems e.g. one question could be "why you need this solution when you already have the mechanism to do the job". During analysis, requirements are cross checked to identify and resolve conflicts. Once the system analyst develops the final SRS document he arranged a session with developer/coder to discuss requirements in detail so to make sure that coder gets it in the right direction. The company B uses prototyping as mean of defects identification during requirements validation to find missing and incomplete requirements. The company also involves customers during this validation process. Besides that they have chat session with clients to make things understandable and clear if they found some ambiguities in requirements. They also don't know about DPTs that are mentioned in section 5.3.

6.1.3 Company C

The Company C provides digital signage solution to its customers in Northern Europe. The organization is expert in software development as well as hardware development. It deals in developing monitors and screens with video interface provided for the digital signage. The organization is the main supplier of digital signage for different retailers and offers its product in almost 30 countries. The organization clients included telecom organization, departmental stores, shopping malls, and restaurants etc in different countries.

6.1.3.1 Interviewee

The interviewee has five years of experience in different engineering disciplines. Currently the interviewee works as an operation manager. The primary responsibility of interviewee is to validate and verify the requirements along with acceptance testing of final system to make sure that requirements are implemented according to requirements described in the SRS. Apart from interviewee's current role, the interviewee was also involved in development activities and worked on different role during his tenure. The interviewee had worked as a developer for implementation of requirements and in quality department as a test to generate and execute test cases based on SRS. The varying and extensive experience of interviewee helped us to get more information about research questions.

6.1.3.2 Defect Types and Their Reasons based on E and A&N

The interviewee reported missing requirements by giving a scenario experienced by him. He also figured out other defect types that originate from E and A&N phases such as unclear requirements, extraneous information, omissions, high volatility rate, inconsistency (not frequent), ambiguity, after releasing product, sometimes they have to do a lot of extra work when they receive a change from the customer. The reasons for missing requirements were due to communication gap between developer and customers and poor requirements analysis while unclear requirements were due to complex nature of features.

6.1.3.3 Techniques Reported by Company C

They use ad hoc approach to analyze the requirements but overall they were fallowing Xp programming approach. When they get basic requirements from the customer about 4 to 5 experts sit together (group meeting) that includes developers and operational manager. The owner of the company has direct contact with customer. He is also involved in acceptance testing. First they try to understand the requirements and do clear all requirements. Then they look at each requirement from design and implementation perspective. If developers find problem or misunderstanding of a requirement they inform their boss (who has direct access to the customers) about that requirement. Their boss in return makes screen shorts or tries to make them understand by sketching diagrams on white board. During this process the inconsistencies and dependencies are discussed and fixed. They use IEEE format for writing SRS. In requirements validation process SRS is sent to customer, tester, designer and operational manager. Tester makes test cases and if find any problem then he informs RE team to correct it or can also give his suggestion. Sometimes they had to develop prototype to clarify a requirements to the customers. When they clarify all requirements then they prioritize the requirements on the bases of experience and value based technique.

When they find a defect during testing phase, they make that defect part of test case document. In this way they can avoid that type of defect for the next time. Further they keep the log history of any missed test case or defect (with causes) that tester finds. This test log is given to developer so that they can know their weaknesses and can avoid making them again. Tester reports most of the defects (more than 80%). They also do not follow any DPTs mentioned in section 5.3

6.1.4 Company D

Company D is a world famous organization deals in mobile and communication devices. The company has its network in almost 160 countries. To deal with customers, company has different Marketing Units (MUs) spread all over the world, who interact with the customers to know their demands and to get features to implement. The MUs then look for requirements and communicated with the development center in Karlskrona where they develop and test the requirements.

6.1.4.1 Interviewees

Two persons were contacted from Company D i.e. one is tester and one is designer. They both interact with the customer's requirements. As described above the MUs department is responsible for gathering requirements where customer give them requirements that are abstract or sometimes they come up with solution as well. They have another department called system handler and system organization who also deals with the RE process. The RE process starts from MUs which act as an interface between company and customers. They gather requirements and the requirements for a particular release end up in their development center in Karlskrona, there are teams responsible for analyzing the requirements. It provides information about the estimation of requirements whether it is possible to implement it or not, how much time they will take, problem with requirements and finalize the SRS. First SRS is produced at MUs for writing the customer's requirements and the final document is produced after analyzing the requirements in collaboration with MUs and development team. Both interviewees have four and nine year of experiences in their related field. They analyze the requirements after getting it from marketing departments. As they have a lot of experience so they analysis process is much more dependent on their personal experience. One of the problems with interviewees is their lack of awareness about technical terms. They just want to discuss about interview questions in their own term which they used within their organization.

6.1.4.2 Defect Types and Their Reasons based on E and A&N

The interviewee's identified unclear requirements as major requirements. They said that requirements are always unclear when they get it from MUs. They are ambiguous and produce different meaning for single requirements. The interviewees as worked in a team discuss and negotiate with the MUs and requirements handler department where they solve the problem. The cause of this problem comes from marketing department where they write the first SRS in collaboration with customer. They also find inconsistencies among requirements due to conflicting requirements. The person who wrote the first SRS made mistakes due to which there can be conflicts or contradiction among requirements. They also experience changes in some requirements even during implementation phase. This will create problems because this can disturb the project schedule. So they are difficult to implement and manage in future releases. These changes are rare as they proceed to development after finalizing the requirements. After analysis of requirements, a final document of requirements is produced and sent back to the customer for final approval in case of changes, they are handled at that level so there is little chance of requirements volatility. Defects are reported by tester most of the time. From the customer`s side, they have configuration problem related to software but this issue arise due to lack of awareness and understanding about products. Defects found fall into different categories on the basis of their severity. The interviewees identified requirements defects like ambiguous, unclear, and missing. Sometimes they are major and sometimes they are minor defects. Sometimes they are not feasible to implement and sometime organization have their own limitation for implementation. They use IEEE standard for specifying requirements. Moreover, there is no classification mechanism for defects instead during testing they defined their own definition to describe the severity of a defect e.g. emergency and intermediate correction etc. During RE process, requirement analysts are given the training about system knowledge but there is no mechanism of explicit RE training. They are depending upon their employee's experience and tacit knowledge but they spend more time to RE process.

6.1.4.3 Techniques Reported by Company D

Although company do not has a specific DP mechanism but they take different approaches at different level of development e.g. pair programming, use of checklist for inspection and design rules before starting development. These rules provide guidelines that act as preventive action. To deal with in-process and customer reported defects, they maintain defect/trouble reports which defined four different levels of defects based on their severity. During requirement engineering process they have some training sessions and courses on methodology that they use i.e. extreme programming. Besides this they are also relying on their experience to avoid defects.

To make the RE process efficient, the company has a separate and dedicated department for gathering requirements from customer. They are continuously intact with the customers in getting new features and requirements all the time. The analysis is performed both by the requirement handler and the design team. Thus there is little chance of lack in requirements in later stages. To find problem in requirements they performed inspection with formal reviews of SRS. They are also not familiar with the DPTs that are mentioned in section 5.3.

6.1.5 Company E

The Company E is a leading Pakistani software organization deals in developing tools for documents composition, resource management, and also provides web-based support for organization to create, manage and optimize their documents work flow. The company develops application for both stand alone solution and integrated solution for existing systems. The organization has customer all over the world in almost 70 countries.

6.1.5.1 Interviewee

Interviewee has been working in software development industry for almost ten years. Mostly he spent time as developer with less customer interaction, complete design and implementation. After getting experience during his job in company E, he was appointed as DM (Development Manager). As a DM he has been involved in communication with testers, requirements analyst, and product manager. Currently he is Project Manager in company E.

6.1.5.2 Defect Types and Their Reasons based on E and A&N

According to interviewee, the rate of defect depends upon the number of customers. The product having greater number of customers have more defect rates. Unclear requirements are identified by development manager. Inconsistencies and ambiguities in requirements are identified by requirements analyst and testers but they are rare (about 5%). They have seen missing requirements after product release. For example when company E launches a product, customer observed its features and finds some missing feature then he inform requirements analyst that your product is okay but if you will add feature X then it would be acceptable for him. This missing requirement is handled as enhancement request.
Since company E has scrum teams (5 to 8 members in each team) for software development and has multiple sprints for each project. Each sprint is consisted of one month or some more days. For each sprint they freeze SRS before implementation. So rate of volatile requirements is very low. Misunderstanding in requirements (20% to 30%) is identified by testers. This misunderstanding in requirements leads the developers to implement wrong requirements.

Some customers don’t like features that company E adds to their products because developers think that customer will love this feature. But on the hand customer complains that it was not his requirement, he doesn’t like this feature. This defect is called gold plating (due to over fleetingness. The interviewee also told that unfortunate Pakistani companies don’t share their industrial research and they do research only to solve their own problems.

The development team most works on finds and fix strategy but according to interviewee, the overall strategy of organization aims at DP which is achieved through testing. Testers ensure that the product is defect free and the defect can not reach customers. This demonstrate that the preventive measures happen only on testing phase

According to interviewee, time parameter is more important for them. They prefer to provide product to the customers on time even if the product has some known defects. By doing so they believe that they gain customer satisfaction and trust because customers might think that the organization has at least deliver something and on time. It means time and customer trust is more important for the organization.

There is also no training mechanism for the employee regarding RE process. The structure of organization promote developers to reach at higher position with respect to responsibility i.e. from develop to requirements analyst or designer.

6.1.5.3 Techniques Reported by Company E

Company E doesn’t have formal process for requirements analysis and they are not using any structural approach. Requirements analysts fallow ad hoc (use their experience and knowledge) approach for analysis. For all requirements, product manager maintains product backlog that can help them in future. He refines product backlog by getting feedback from development manager. Developers are involved in requirements validation process. They get SRS and analysis each requirement, if they find that requirement X is not feasible for

implementation or something is not clear then they send their feedback to requirements analyst. If unclear requirements remain persistent then requirements analyst use a method that is called POC (Proof of Concept) to clear the requirements. POC is implemented by developing a prototype that shows the real meaning and picture of unclear requirements to the developer. Requirements analysts are expert in developing prototype for a particular requirement. Requirements analyst can also communicate with business unit, if some problems regarding requirements need customer involvement. In most of the cases development manager has to consult with requirements analyst about problems in SRS
Business development management team is spread in different parts of the world. They survey the market and looks for any business opportunity (what products are going on, who is looking for solution for their product) for their organization. After getting an opportunity, they try to sell their product if it provide good solution otherwise they enhance their product to fulfill customer requirements. In other words business units is the part of the requirements elicitation process business units has direct communication with product manager and requirements analysts. They are not familiar with the DPTs that are mentioned in section 5.3.

They don't have a formal process for DP. They develop a document named problem log. The problems that they face throughout product development are logged in the form of a document. These problems are considered for the next just avoid them to re occur. This problem log is used during testing and automated testing process. Before starting a project they specify all areas where they need more concentration. They give importance to a customer if he is main figure head.

6.1.6 Company F

The Company F is a Swedish based software organization that provides software to improve business performance for telecom organization. The organization develops software for different telecom operators where they manage and optimize their Business Operation Systems (BOS) consisted of Business Support Systems (BSS) and Operations Support Systems (OSS). As there is a wide range of voice traffic exchanged between telecom carriers, so there is need of support tool for handling the trading and voice traffic among them. The organization provides this facility through their tools called Carrier Cockpit Suit, for telecom operators. Different telecom operators are the sole customers of organization. The organization do not have a dedicated department for managing customers requirements instead they have a support department which is responsible for providing services to organization existing customers who put problems in existing systems and suggestion for improvement in the system.

6.1.6.1 Interviewee

The contacted person from Company is the product manager who is responsible for defining features for a product release. The interview has many year of experience in his field and directly deals with defining requirements, set plans for different product releases. The interviewee directly deals with customers who are representative of different telecom operators who put different requirements on system. The product manager along with team gathers and manages requirements for these customers. This product management team set discussion with customers in order to identify their needs. At high level, customers create a short abstract definition of what they want. The interview is a technical person and responsible for writing use-cases for functional requirements for the system. These use-cases are further analyzed by product management team who resolve conflicts among requirements prioritize and estimate them. The organization uses its own template for specifying requirements which is based on UML notation. The interviewee is also responsible for validation requirements along with expert consultant. The varying responsibilities of product manager were helpful for data collection and through his experience it was beneficial for us to find relevant and related information about research questions.

6.1.6.2 Defect Types and Their Reasons based on E and A&N

During requirements analysis sometimes they find use cases really not understandable, inconsistent and ambiguous. They believe that requirements don't have frequent changes and SRS remains stable and they don't have volatility factor so serious because they freeze SRS before implementation of requirements. During testing phase they mostly identify defects that are related to missing requirements (time plan for requirements validation, late involvement of tester in test specification are reasons), and defect related to implementation. After putting their product into operation, customers also report some defects that are related to requirements. Sometimes they have to face missing non functional requirements defect. For example interface that they develop don't satisfy the customer. In response they have to do rework and sometime they do major rework to satisfy the customer. If inexperience people have developed a product then customer complains major defects (ambiguous requirements, wrong requirements) that cause tremendous rework but it happens in rare case. Summary of most common requirements defect types reported by industrial study is given in table 6 (see section 7.1). Definition of major and minor requirements defect is given in RQ 1.3 in section 7.1.

6.1.6.3 Techniques Reported by Company F

Company F don't have dedicated department for the RE process. A dedicated support team is available which directly deals with customers. The responsibility of this department is to deals with customer complaints about the products or improvements in the products. After getting requirements they write short description of all requirements. At product management level they have some expert consultants where they look, defined and describe how they do realize actual requirements. Here they estimate the requirements and after that they communicate back to the customer where they look for their solution. They don't use any formal language to write SRS but only use cases and plain English and try to make it very simple. They avoid complex English word and phrase. They draw uses cases to model the functional requirements from short description of requirements and these uses cases are reviewed by product management team, developers and expert consultants. They use pear reviews and in some case for more complex requirements they use prototyping technique. Some time there are lots of inconsistencies in use cases and they go back to customers for negotiation.

They believe that for complex RN process is very time consuming and that's why have to use prototyping technique to make the customer understand a requirement. To avoid inconsistency, ambiguity and misunderstanding in uses case they try to make use cases as simple as possible. During requirements validation they make real life scenarios and test cases from SRS and customer is involved only when he request change. They validate final SRS in a formal meeting that is attended by project manager, product manager, developers and testers. They do have open discussion in this meeting where everyone is freely allowed to share his views. They are not familiar with DPTs that are mentioned in section 5.3.

As a DP measure they document all defects with scale of major and minor defects. The major defects are avoided in early stages from reoccurring. This process of DP is handled by QA department and interviewee doesn't know about this process. They use a tool named Acura by which they keep track of releases, software cycle, and software release changes.

7 DATA ANALYSIS

Authors have conducted both empirical and literature research to support the aim of the thesis. Throughout this research authors have used qualitative research methodologies like literature reviews and industry interviews. Each RQ was achieved by an appropriate research methodology. The detail of appropriate research methodology for each research questions and corresponding objective and expected outcome achievement is described in table 1.

The results of empirical and literature research is shown in chapters 5 and 6 in detail. On the bases of results authors have proposed some solutions that ultimately answer one of the given RQs. In this chapter authors have conducted an analysis of the study findings. This will helps authors to determine that to what extent these findings are helpful in meeting all RQs (that will be done by mapping RQs with study results), and will discuss some interesting issues based on data that has been collected from literature and industry.

7.1 RQ1 & Data Analysis

RQ1 has been divided into three sub questions (RQ1.1, RQ1.2, and RQ1.3). The answer of RQ1 will be achieved by explicitly answering subsequent sub questions. RQ1 is given below

RQ1: What are requirements level defect types and reasons for defects related to E and A&N phases that can cause major rework in later stages of SDLC?

RQ1.1 - What are the most common defects types reported by research based on E and A&N?

To identify most common requirements defect types reported by literature research, a thorough literature review (see section 5.1) has been conducted that kept focus on industrial case studies and industrial experiments so that authors can find a realistic results about requirements level defects that originate from E and A&N phases of the RE process. The summary of literature review is given in table 4 that shows mostly occurring requirement defects, their description and number of references from literature (industrial experiments, case studies).

Authors have found lots of defects types that are associated with software requirements and they have put all these defect types in a single category called requirements defect types as shown in column two of table 4. Each defect type has been reported by multiple sources (i.e. case studies, industrial experiments).

Table 4: Most common requirements defect types based on research from literature

S.No.	Requirements Defect Types	Defect Types Description	Ref.
1.	Ambiguity	It is interpretation of requirements in more than one way thus creating different understanding of a requirement. So ambiguity leads to confusion and extra rework during software development process. For example if requirements validation process was not performed well then ambiguous requirements will be put in design and implementation phase. The tester could have another interpretation of that requirement and he will confuse about what he should test. So the cost and effort spent during design and implementation phase would be wasted if customer had different interpretation of that ambiguous requirement.	[25, 4, 31, 20, 35]
2.	Inconsistencies	Inconsistency means contradiction in requirements [20] or requirements conflicts with one another. For example a user wants access to a particular portion of database but database administrator does not grand this access due to security policies. So a conflict arises between these two requirements of different users.	[24, 26, 20, 35]
3.	Omissions	Omission means something that was left out or requirements that was not written down. For example if an ATM machine does not handle multiple cards simultaneously, it means this functionality was omitted (was not written or was skipped).	[24, 2, 25, 4, 26, 30, 20]
4.	Unwanted/ unnecessary	This type of defects includes some functionality or features that customer does not want but the developers add it just to make the customer happy. Developer thinks that customer will love it. In other words the developers perform Gold Plating. For example if customer did not mention the he want a highly graphical interface but the developer adds it just make the customer happy.	[2, 4, 34]
5.	Volatility	Volatility is the tendency of requirements change over time. If RE process will continue even during implementation, then requirements will becomes more volatile. In other words the chance of considering a requirement for current release becomes minimal in later stages of development cycle. If this volatile requirement is made part of current release then it might take more cost and time.	[38, 39, 31, 34]
6.	Clarity	Requirements description may not make clear that what is required actually	[26, 30]
7.	Incorrect fact	A requirement fact that cannot be true in particular condition of the system, wrong behavior),	[22]
8.	Missing Requirements	Missing functionality/feature, Missing interface, Missing performance, or Missing environment)	[24, 25, 4, 26, 30, 31, 20, 34, 35]
9.	Extraneous Information	Unnecessary or unused information	[22]
10.	Others	Problems or defects that cannot be put into rest of defect types that are mentioned above	[25, 26]
11.	Unstable requirements	Software requirements become unstable when changes in SRS occur frequently.	[37]

In table 4 the authors have listed requirements defect types (with their description) reported by the literature study and this table has been extracted from the most relevant research articles that were 17 in numbers. If we consider these 17 articles as sample from academia then we can figure out the most common requirements defect types reported by literature. Table 4 shows that missing requirements defect has been reported by 9 articles, omission by 7 articles, ambiguity by 5 articles, inconsistency & volatility by 4 articles, unwanted/unnecessary & clarity by 3 articles, and incorrect fact & unstable requirements by 1 article. On the bases of preceding calculations we can say that missing requirement, omissions, and ambiguous requirements are the most commonly occurring defect types. Inconsistency, volatility, unwanted, and clarity are relative less commonly occurring defect types. The following paragraph contains some references from literature study that are in the favor of continuing paragraph.

It has been noted that information obtained from some articles like [24, 25] clearly mention the most important defect types that hold major contribution among all types of defect types at RE level. According to [24] about 83 percent of defects are due to incompleteness, missing, omitted, and incorrect requirements defect types and in [25] it has been reported that omitted defect type hold about 56 percent of requirements defects and , 17 percent defect are due to ambiguous requirements. An experimental study in [20] has also put missing information and wrong information on top priority. In [31] a case study was conducted where they found that major requirements defect types are missing requirements, ambiguous requirements, and volatile requirements. In another case study it was proved that most of the defects were due to missing, incomplete, tacit, assumed, and volatile requirements in the implementation phase [34].

From above discussion in last two paragraphs, it has proved that requirements defects like missing, omissions, ambiguous, and volatile requirements are the most common requirements defect types reported by literature.

The most common defect types based on research from literature is described in detail in section 5.1. The study results are based on research from different case studies carried out to find the requirements level defects. The defects found out in literature research are summarized in table 4. Based on literature study results in section 5.1 it is found that;

1) Not all defects are originated from a single phase of RE
2) Defects originated in one phase are detected in later phase i.e. during analysis and validation process.
3) Not all defects hold the same level of importance i.e. some defect types are more critical while others are less serious

Requirements level defects are originated and treated at different level of RE. They fall into different categories based on the phase where they are initiated and detected

1) Elicitation related defects - Elicitation related defects originated as a result of problems in;
    - Communication between customer and requirements analyst,
    - Technique (s) used for elicitation
    - Domain knowledge
    - Users/customers involvement
    - Requirements analyst experience

Typically defect types at this level includes

- Unnecessary or unwanted requirements – these requirements are added by the requirements analyst which resulted to gold plating
- Tacit requirements – requirements that are in the mind of analyst, although these requirements are elicited but they are in the mind of analyst
- Lack in this phase leads to volatility in requirements. If the elicitation process in not properly performed, the system may ends up with wrong and unexpected results.
- Missing information – Requirement that are not elicited

2) A & N related defects are
    - Inconsistency

- Incompleteness
- Feasibility

3) Specification related defects
    - Ambiguities
    - Typographical error
    - Input/output mismatching
    - Missing and incorrect data
    - Poorly written requirements
    - Minimal requirements
4) Validation related defects
    - Missing information
    - Incompleteness

There are some risks that are associated with the RE process (see section 5.1 and table 5). These risks can play awful role in the origination of defects associated with the requirements. So measures should be taken to avoid these risks. Authors have given more detail of these risks and their consequences in table 5 given below.

Table 5: Risks related to RE process leading to requirements defects

Risk	Description	Defects	Consequences
Insufficient user involvements	Minimal users involvement in elicitation and negotiation phase lead to missing, or incomplete information	Missing information Incomplete requirements	Lack of user involvement leads to lack in requirements. The consequences of this could be unrealistic system.
Gold plating	Adding an attractive feature just to make the customer happy	Unnecessary requirements, unwanted requirements	Wastage of resources in term of time and human
Creeping users requirements	Too many change requests by users in requirements	Requirements volatility Unstable requirements	Too many change request make the requirements unstable enough to increase project time
Ambiguous requirements	Confusing and unclear requirements many lead to different interpretation of requirements which is difficult to implement and test	Unclear requirements Confusing requirements Requirements misinterpreted differently	Implementation is different than specified in specification or the requirements described in specification in a different way that produce different meanings from actual context of the requirements
Overlooked user classes	There different classes of users that use different subsets of features; have different frequencies of use, or different experience levels. They should not be overlooked by requirements analysts	Missing and incomplete information, inconsistencies	It may leads to dissatisfaction of some users of the product
Incorrect planning	Inaccurate estimations are done like cost estimation.	Infeasible requirements Compliance	Poor cost estimation leads to frequent requirements changes, missing requirements, insufficient communication with users, poor specification of requirements and insufficient requirements analysis.

RQ1.2 - What are the most common defect types and reasons for defects originating from E and A&N phases as reported from Swedish and Pakistani software companies respectively?

Second and third parts of RQ1 is related to empirical study that was done by conducting industrial interviews in both Pakistani and Swedish software companies (see section 6). The detailed description of defect types and their reasons is given in appendix E that will help the readers to clearly understand them.

On the bases of empirical study results, the RQ1.2 requires aggregation of some interesting data categories like requirements defect types, their reasons, and name of companies (from Pakistan and Sweden) that report requirements defect types and their reasons. The authors have collected preceding data from three Swedish and three Pakistani software development companies during interviews. During interviews the authors wanted to explore two more data categories like rate of defect occurrence and their severity level because these factors will help in explaining the importance of requirements defect with respect to rework in later stages of SDLC. This aggregation of important points is given in table 6 given below. The purpose of this table is to explain the summary of industrial research that particularly covers RQ1.2.

Table 6: Most common requirements defect types based on research from industry

Pakistani Companies = B, C, E
Swedish Companies = A, D, F

Serial No.	Defect Type	Defect Reasons	Rate of Occurrence	Severity Level	Reported Company
1.	Missing	Limited time allocated to RE process, lack of domain knowledge, communication gap between customers and developers, poor requirements analysis, time plan for requirements validation, late involvement of testers in test case specification process	High Low (A)	Major Minor (A)	A, B, C, D , E, F
2.	Unclear	Requirements are not initially clear, lack of investor involvement, complex nature of requirements feature, wrong interpretation	High	Minor Major (D)	A, B, C, D, E
3.	Missing Feature	Tacit or unwritten requirements	Medium	Major	B
4.	Inconsistency	Disagreement among customers, conflicting requirements, bad requirements writing ability, lack user involvement	High Medium (C) Low (E)	Minor	B, E, C, D, F
5.	Extraneous Information	Unnecessary customer's needs	Medium	Minor	C
6.	Ambiguity	Poorly written SRS, complex feature, wrong interpretation or misunderstood requirements	Low (E, F)	Major	C, D, E, F
7.	Volatility	Changing work environment, organizational complexity and conflicting requirements	High (C) Low (A, B, E, F, D)	Major	A, B, E, F, D, C,

8.	Unstable Requirements	Frequent change requests	Low	Major	C
9.	Omission	Tacit or unwritten requirements	Medium	Major	B, A, C
10.	Misunderstood Requirements	Misunderstanding in requirements, Remote location communication, Culture, political environment, Involvement of inexperience people	Medium Low (F)	Major	E, F
11.	Unwanted/ Unnecessary	Gold plating, over fleetingness	Medium	Major	E
12.	Missing quality attributes	time plan for requirements validation, late involvement of testers in test case specification process	Medium	Major	F

During literature study authors found some interesting points regarding RQ1 (i.e. volatility and instability in requirements) that seem to be neglected by literature (reported by only 4 articles out of 17, see table 4). Unstable requirements are associated with the frequency of change occurrence in SRS and volatility depends on time when a change occurs. For example if a change occurs in RE or design phases of SDLC then requirements are less volatile and on the other hand requirements become more volatile as the changes occur during implementation, testing and maintenance phases of SLDC. It means volatility of requirements is directly proportional to time of development process.

Volatile requirements are sometimes called the cost drivers because they demands extra cost and effort. They also affect the defect density in code phase, quality of code, quality of project management and quality of developer's capability. An unstable SRS is very easy to handle in early stages of SRS but as the time passes it becomes very difficult to manage it because the factor of volatility becomes active in later stages of SDLC and this situation can cause tremendous rework.

RQ1.3 - Find possible rework caused by each defect?

Sub question RQ1.3 deals with the severity of industry reported defects in term of their major or minor rework. This information is also described in table 6 against each defect types under column "Severity Level". This sub question describes the rework caused by each defect if it would be identified and fixed in later stages of SDLC (implementation, testing, and maintenance). In order to find the possible rework caused by each defect types, the authors have provided a definition of major and minor defect based on rework they caused. This definition was presented to the interviewees during interviews to find the answer of this sub question. The definition of major and minor is given below.

- Major defect is one which can cause a major rework when it would be identified and fixed during implementation or after product release.
- Minor defect is one which has no significant rework when it would be identified and fixed during implementation or after product release.

Table 6 shows that ambiguity, omissions, missing requirements (missing feature, missing quality attributes), unnecessary/unwanted requirements, volatile requirements, unstable requirements, misunderstood requirements, are reported major defect types by interviewees. On the other hand inconsistency, extraneous information, and unclearness have been reported minor defect types by most of the companies (both form Pakistan and Sweden, see figure 12).

Since the aim of this thesis is to reduce the rework in later stages of SDLC that is caused by defects originating from E and A&N phases; so our focus has been only major requirements level defects because prevention of these major defects will ensure reduction in rework in later stages of SDLC.

7.1.1 Summary of Most Common Defects Types and Their Causes that Originate from E and A&N Phases in BESRE

Based on findings from above sub questions, the complete answer of RQ1 can be found by combining table 4 and 6 in a separate table. The results from both empirical and literature studies are summarized in table 8 that is given below. Table 8 answers the questions like what are most common defects types and their reasons that originate from E and A&N phase, how much they are severe (major or minor) with respect to rework, and what is their rate of occurrence. It also explains that which company is reporting defect types, and what is severity level of a defect with respect to that company?

Authors have observed during interview that most of the companies freeze SRS before starting the implementation process. If they have new change request during implementation or after implementation then they put it in next release. However if the customer insists the project manager to add current new requirements in developing project then it might need marvelous amount of rework in the form of changes in software architecture, design and implementation overhead. Authors think that it is not a good practice for project based companies to put volatile requirements in next release because customers are not always in tradeoff position. So there is need to overcome the factors that cause volatile requirements like changing working environment, organizational complexity, and conflicting requirements.

According to IEEE a good SRS has some attributes like unambiguous, complete, verifiable, consistent, modifiable, and traceable [77]. Authors have noticed that defects that cause major rework in later stages, were not related to preceding attributes of an SRS but it were related to omissions, missing requirements, unclear requirements and so on.

Poor requirements are also a big source of project failures and partially succeeded in the form of partial functionalities, major cost overruns, and significant delays. An empirical study was conducted based on large survey where authors found that major reasons of poor requirements are the lack of user involvement (13%), requirements incompleteness (12%), changing requirements (11%), unrealistic customer's expectations (6%), and unclear objectives (5%). So changing requirements has overwhelming percentage (11%) of project significant delay, incomplete functionalities, and major cost overturns [30]. Now we will see what defect types (described in table 6) have reasons similar to preceding defect reasons. Table 7 is extracted from table 6 and from discussion above. It shows that defect types like extraneous information, unstable requirements, and missing requirements have significant impact on project failure or success because these defect types can lead to delayed, incomplete and a more costly project.

Table 7: Requirements defect types and their reasons that affect project plan

Serial No.	Defect Types	Defect Reasons
1.	Extraneous Information	Customer's expectation
2.	Unstable Requirements	Changing requirements
3.	Missing Requirements	Requirements incompleteness, lack of user involvement

Table 8: Most common requirements defect types based on research from literature and industry

Pakistani Companies = B, C, E
Swedish Companies = A, D, F

Serial No.	Defect Type	Defect Reasons	Occurrence Rate w.r.t Companies	Severity Level	Reference
1.	Ambiguity	Poorly written SRS, complexity Of features, misunderstandings or wrong interpretation	Low (E, F)	Major (C,D,E, F)	[25, 4, 31, 20, 35]
2.	Inconsistency	Disagreement among customers, conflicting requirements, bad requirements writing ability	High (B,D,F) Medium (C) Low (E)	Minor (B, C, D, E, F)	[24,26, 20, 35]
3.	Omission	Tacit or unwritten requirements, skipped requirements	Medium (B, A, C)	Major (B, A, C)	[24, 2, 25, 4, 26, 30, 20]
4.	Missing	Limited time allocated to The re process, lack of domain knowledge, communication gap between customers and developers, poor requirements analysis, , lack user involvement, time plan for requirements validation, late involvement of testers in test case specification process	High (B, C, D, E, F) Low (A)	Major (B, C, D, E, F) Minor (A)	[24, 25, 4, 26, 30, 31, 20,34, 35]
5.	Unnecessary/Unwanted	Gold plating, over fleetingness	Medium (E)	Major (E)	[2, 4, 34]
6.	Volatility	Changing work environment, organizational complexity and conflicting requirements	High (C) Low (A, B, E, F, D)	Major [38]	[38, 39, 31, 34]
7.	Unclearness	Requirements are not initially clear, lack of investor involvement, limited time allocated for RE and for validation purposes complex nature of requirements feature, wrong interpretation	High	Major (D) Minor (A, B, C, E)	[26, 30]
8.	Extraneous Information	Unnecessary customer's expectations	Low (C)	Minor (C)	[22]
9.	Missing feature	Tacit or unwritten requirements	Medium (B)	Major (B)	[20]
10.	Unstable Requirements	Frequent change requests		Major [37]	[37]
11.	Misunderstood Requirements	Misunderstanding in requirements, Remote location communication, Culture, political environment, Involvement of inexperience people	Medium (E) Low (F)	Major (E, F)	
12.	Missing quality attributes	Time plan for requirements validation, late involvement of testers in test case specification process	Medium (F)	Major (F)	[20]

Table 8 shows some defects types that have high rate of occurrence like inconsistency, missing, omissions, incorrectness, and unclearness. The preceding defect types (accept inconsistency) are also major defects with respect to cost and effort required when they would be found and fixed in latter stages of SDLC. Here we can raise a question that how much percentage (of total requirements defects found) these major defect types hold? It is also clear from [24] to find answer of this question where author have conducted an experiment to find requirements defects which includes incompleteness (20.9%), omitted/missing (23.9%), and incorrect (23.9%) accounted for almost 80% of requirements defects

7.2 RQ2 & Data Analysis

RQ2 - Are there any differences between Pakistani and Swedish companies in term of types and rate of defects originating from E and A&N phases?

This RQ is based on empirical study described in sections 6 and it is found that there is a very slight difference between defect types and rates of defects between Pakistani and Swedish companies. The defects and their rates vary and depend upon the RE process they follow, techniques and methods used during RE process, domain, company size, and time and resources (people) allocated to the RE process. To know the difference between these two countries in term of defects and rate of defects, authors have examined them individually (within countries) and results are compared with each other (across countries i.e. Pakistani vs. Swedish companies) to find interesting concerns (similarities and dissimilarities).

7.2.1 Comparison between Pakistani Companies (B, C, E)

Defect types reported by Pakistani companies are more in number as compared to Swedish companies. Requirements are unclear (company B and E) most of the time because of lack of investor involvement, limited time allocated to RE and other political factors involved within the organizations. They reported missing features and missing functionality and they are mostly reported by customers (company B). According to Pakistani companies this defect types of missing requirements are hard to fix if they are related to functional requirements of the system. Defects are mostly reported by customers and according to interviewee sometime they are greater than 60% (company B). They also reported omissions, extraneous information, inconsistencies, conflicts (rare), misunderstanding (20 to 30 % reported by company E) and ambiguous requirements (5% to 30% reported by company E). Like Swedish companies, they also have low volatility rates, although requirements do change during implementation but they are very rare.

The above mentioned defects vary among organizations. They also focus on identifying defects at requirements level and during testing phase. According to some organization, more than 74% (company B, this percentage is about defect finding only in RE phase) defects are identified and fixed during RE process. Defects reported during testing are estimated to 60% to 70% (B & C) and rest of the defects is reported by customers after release which is more than 60 % (company B)

The authors have created a table that contains discussion of some interesting data categories. This information is given in table 9 given below.

Table 9: Comparison of Pakistani and Swedish companies with respect to defect reporting, RE process, SRS standards, and DP approaches

Company	Mostly Defects are Reported By	Requirements Engineering Process	Usage of IEEE Standard for SRS	Defect Prevention OR Defect Find & Fix Approach
Pakistani Companies				
Company B	Most of the defects were reported by end-use in the form of missing features (about 60 percent of all defects)	They give limited time to RE process, have dedicated RE department, but do not have proper training	No	They believe in DP strategies and approaches
Company C	Mostly by software testers that is around 80 percent	No training for RE process	Yes	They believe in DP but seems that they have more focus in finding and fixing in testing phase
Company E	Software testers and sometimes by customers (due to gold plating) but it is rare.	No training for RE process , informal RE process	No	They believe in finding and fixing of defects at testing phase
Swedish Companies				
Company A	Testers during implementation	Informal training for RE process , no dedicated RE department	No	DP approach
Company D	Most of times software testers find defects related to requirements and implementation	Dedicated RE department, but no explicit training for RE process and depends upon employees experience	Yes	They believe in DP approach
Company F	Software testers and sometimes by customer but it is rare.	No dedicated department for RE process but a dedicated team is responsible for support customers about improvement in product and handles customer complaints	No	They believe in prevention of defect in early stages by using a software tool

This table shows that

1) Company B and C believe in DP but unfortunately they don't have any proper DP technique or method that the authors have mentioned in table 10. The reason behind it is the unfamiliarity of companies with DPT and DP methods. On the other hand company E pays its attention on finding and fixing defects at software testing phase. They spend the major part of their effort and resources on testing phase to find all kinds of defects. It is also reported by literature that most of the organizations believe in finding and fixing defects in later stages of SDLC [1]. When authors asked interviewee that why not requirements analyst spend more time on requirements analysis to find defects then interviewee replied that our requirements analysts are very busy persons and they spend

their precious time on other activities. He further added that the software testing department is responsible for finding all kinds of defects.

2) In company C & E most of the defects are reported by the software tester. It means they don't spend more time and effort on requirements analysis process, and they don't use any formal DPT. One more reason for it is that they are depending only on developer's personal experience and knowledge. On the other hand interviewee from company B told the authors that around 60 percent of defects are reported by customers in the form of missing features. It shows that company B doesn't have good enough software testing process and we know that finding and fixing software problems after product release is 100 times more expensive than finding and fixing during RE practices or during design phase [19, 42]. Further If requirements defects would mistakenly allowed reaching the customer or tester then these defects would be very costly to fix in the form of tremendous rework [4] So company B would have to spend more time and effort for each defect correction. One more reason for it is that company B spend very limited time for RE process during software development process.
3) It is noted that not a single Pakistani company arrange training for their RE process (give a ref here to reason preceding paragraph). In company E the senior developers are eligible for requirements analyst post and in this way they don't have need to hire special software engineering.
4) SRS is very important RE work product because it defines the needs and boundaries of the software product. It also acts as an agreement between customers and developers. It plays pivotal role in the success or failure of the developing product [19]. According to IEEE a good SRS has some attributes like unambiguous, complete, verifiable, consistent, modifiable, and traceable [77]. It is noted that only Company C uses IEEE standard for SRS creation but company B and E don't use it. It means company C can create a quality SRS than companies B and E.

7.2.2 Comparison between Swedish Companies (A, D, F)

The comparison of Swedish companies based on table 9 shows that

1) All three companies believe in DP in early stages but don't follow any formal DPT or DP method as describe in table 10. It has same reason like Pakistani companies that is unfamiliarity of companies with DPT and DP methods.
2) In companies A,D, and F the software testers reports major part of all types of defects. During requirements analysis they correct some defects found in requirements but their focus remains at software testing phase to find and fix all kinds of defects. It means they don't spend more time and effort on requirements analysis process, and they don't use any formal DPT. One more reason could be that company A & F don't have a dedicated RE department but company B and D have a well established RE department consisting of well experienced requirements analysts.
3) Only company D was using IEEE standard for SRS development and other two companies has their own standard. So attributes of quality SRS like unambiguous, complete, verifiable, consistent, modifiable, and traceable cannot be met without using IEEE standard for SRS development as described in [77].

Company D and F face less defect types in development and low defect rates after releasing their products. The organization (D) uses extensive testing in order to deal with defects before release. All of these Swedish companies are not aware of DP techniques rather they are using defect identification technique during informal requirements inspection with ad hoc (company A) and checklist based (company D) reading techniques.

Mostly reported defect types from Swedish companies are missing, inconsistency, unclear and ambiguous requirements (company A, D, and F). The reasons for these defects are the less involvement of design team with the marketing department and less interaction with the customers (see section 6.1.1, 6.1.4, and 6.1.6). The major issue in this regard is the communication issue and less customer involvement. However besides these defects they have also reported missing non-functional requirements, inconsistent and conflicting requirements. Changes are very rare in these companies and their rate is very low. They are managed at requirements level and there is very little chance of volatility in later stages because they freeze SRS before implementation and any major changes in requirements are put in to next release.

As described before that these companies believe on reactive strategy of finding and fixing defects. So they mostly uncover defects during testing phase before release (Company D, F and A). It is also observed that the rate of defects found after release is very low for the organization which has a dedicated RE department (company D).

The customers report defects are classified based on organization self defined criteria (company D and F) which describes the severity and criticality of those defects. These defects are prioritized based on classification schema (i.e. most critical, critical, broken functionality etc.) that described which defect is most important and should be considered and solved first. Out of three Swedish companies it is found that they are not familiar with DPT's (company A, D, and F.

7.2.3 Comparison between Swedish and Pakistani Companies

There are some more interesting points noticed by the authors during data analysis that have been discussed below,

7.2.3.1 Use of Defect Taxonomies

It is noticed that out of six companies not a single company is using defect taxonomy mentioned in section 5.2. They are not widely using the structured defect taxonomies like Boris Beizer and ODC. Some organizations have their own defect taxonomy (like company E) as defect preventive measures .on the other hand the authors could not find even a single defect taxonomy that focuses only on requirements defects. For example Boris Beizer defect taxonomy classifies defects of requirements, design and implementation level. It is very important to pay more attention on requirements defect taxonomy because it any measure that will be taken to stop requirements level defect from penetrating into later stages will save effort and time. We know that if requirements defects would mistakenly allowed reaching the customer or tester then these defects would be very costly to fix in the form of tremendous rework [4].

7.2.3.2 Root Cause Analysis

One very important practice that has been neglected by most of companies (both Pakistani and Swedish) is root cause analysis. During the interviews authors have found that companies don't want to spend time in formal root because analysis of defects found during requirements analysis and software testing processes. They keep focus on identification and fixing the defects or faults. Root cause analysis is very important because it inform the software development team that what is the category of cause of error (like communication, oversight), how and at what stage error was introduced, and how can we prevent this error in future. In similar type of projects the root cause analysis can play a vital role in removing major defect types that cause major rework in later stages of SDLC. [1] It is very similar with real life example that if a patient suffering from fever comes to physician who recommends him to take medicine that cure his fever only. If physician would not try to diagnose root cause of the fever (it might be chest inflammation, infection in small intestine, or infection in tonsils) then patient would suffer from the fever again.

7.2.3.3 RE Risk Consideration

According to [4] there are seven requirements level risks that can have bad affect on requirements if they would happen. These risks are explained in section 5.1 and table 5. Most of the risks are related to elicitation phase of RE that directly related to customer or users. During industrial interviews the authors have noticed that most of companies are not taking care of these risks and merely depending on their experience and knowledge (this practice is high in Pakistan than Sweden). They pay attention to a customer if he is the main financing figure head and their developers sometimes do gold plating (company E). Most of companies have revised that they believe in catching defects as early as possible but they don't have any formal method or DPT that you can see in table 10. Some interviewees (from company A and C) showed their will to improve RE process but some were satisfied with their RE process such as from company E, and B. Here we can say that if companies get satisfied with their informal RE process and carry on finding and fixing requirements defects in later stages of SDLC then it could be one of the reasons to get 50 to 70 percent defects form requirements and to spend 40 to 50 percent of the whole budget on avoidable rework as mentioned in [2, 3, 4, 51].

7.2.3.4 Awareness of DPTs and DP Methods

Authors have also noticed that companies (both Pakistani and Swedish) even don't know about the names of DPTs and methods that can be found in literature. In the light of preceding discussion it is recommended that companies should keep in touch with current research so that they can find solution of their problems that cause tremendous rework down the software development road.

7.2.3.5 Acquisition of Academic Research

One of our interviewee (company E) told us that Pakistani companies don't share their industrial research in open market. They use their research for their own benefits but authors think that sharing of mutual problems and their solutions among Pakistani companies can solve industrial problems more rapidly. It is noted that companies those deal with similar types of projects have less rate of requirements level defects (company E) and they spend less effort and time in later rework.

7.2.3.6 Training for RE Process

During interview the authors found that Pakistani companies don't hire trained, well educated and specialist requirements engineers. They appoint a product manager who is most senior developer or tester (see Company B & E). Even they don't have special training for new requirements engineers. On the other hand some Swedish companies have training for new requirements engineers like company A (informal training). Company D is even hiring requirements engineers who are specialist in RE with related educational background.

7.2.3.7 Number of Developers for Each Project or Release

It was noted during interviews that companies were appointing not more than 15 to 18 people for single release or project. For example in company A there were three scrum teams having maximum 5 people in each team, and in company E there were also three teams having 5 to 8 people in each team. We also know that each inspection team requires 3 to 5 people.

7.2.3.8 Defect Types Comparisons

Table 6 shows that defect types having high severity and high rate of occurrence is only missing requirement and it is reported by all six companies from Pakistan and Sweden. Some major defect types have medium rate of occurrence like omissions, misunderstood requirements, unwanted/unnecessary, and missing quality attributes. On the other hand some defect types have low rate of occurrence but severity level is high such as ambiguity, and volatile requirements. The defect types having high rate of occurrence and minor severity level are inconsistency, and misunderstood requirements, extraneous information has low rate of occurrence and minor severity level, and unclearness has high rate of occurrence but minor severity level with respect to rework in later stages of SDLC. All preceding severity levels and rate of defect occurrence were reported by both Pakistani and Swedish companies as shown in table (combined). The summary of continuing paragraph has been explained in figure 12 that is given below.

It is clear from figure 12 that missing quality attributes unwanted/unnecessary requirements, misunderstood requirements, omissions; missing functional requirements are the most common and major defect types. It means that more effort and time should be spent on these defect types to minimize avoidable rework that is done when preceding defect types are identified and fixed down the development road. On the other hand volatile requirements, and ambiguous requirements defect types are major but happening rate is low. We can also pay attention to volatility and ambiguous requirements because they also major and can cause tremendous rework in later stages of SDLC in the form of more time and effort.

It has been proved from discussion (see below table 5) that missing, omissions, ambiguous, and volatile requirements are the most common requirements defect types reported by literature. The comparison of red circled requirements level defect types (see figure 12) with defect types reported by literature, proves that the Union (mathematical binary operator) of both provides the same list of major defect types that has been circled with red color in figure below. So we can say that figure 12 shows the most common and major requirements level defect types reported by industry and academia.

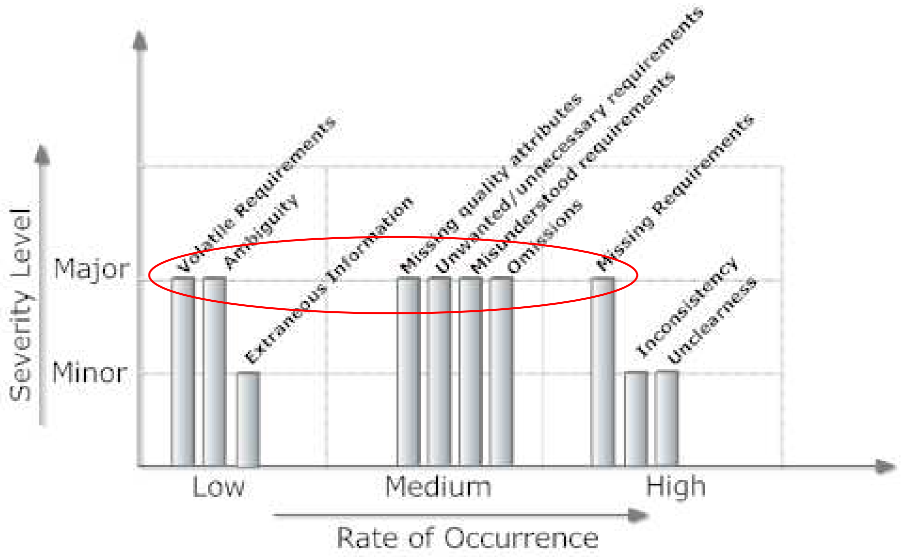


Figure 12: Common and major requirements level defects types reported by industry and academia

Table 6 contains four important defect parameters such as defect types, defect reasons, rate of occurrence, and severity level. High defect occurrence rate means high priority to prevent it from occurring again. However it does not mean that every defect that happens frequently causes major rework. For example inconsistency in requirements has been reported with high rate of occurrence but it is minor with respect to rework if it occurs in later stages of SDLC (see table 6). This study mainly depends on severity level and partially depends on rate of defect occurrence. To fulfill the purpose of this study all major defects having high rate of occurrence have been put on top priority and is recommended to spend more time and effort to prevent them from reoccurring. There are lots of defect types that have been identified so for (see table 6) but all are not major (see definition of major and minor defects in above paragraphs). So instead of spending time on all defects types in the RE process we can spend time and effort on only major defects that cause major rework in later stages in the form of extra time, cost and effort.

One thing should be noted that the authors decide, any major or minor defect type by considering number of companies reporting that defect type. It means that the authors democratically decide major or minor defect types and severity levels (low, medium, high). For example the severity level of defect type inconsistency is reported high by companies B, D, E, medium by company C, and low by company E. In this way collectively inconsistency will be considered as high in severity level.

7.3 RQ3 & Data Analysis

RQ3 - What DPTs is associated with each defect type that is being practicing in industry based on E and A&N?

Preceding RQ provides complete information regarding defect types originating from E and A&N phases of RE. The requirements related defects data demonstrated above provide useful information to learn about the defect types, reasons of those types and their severity in term of rework in SDLC. Besides that there is need to know and learn about the underlying techniques or methods found in both academia and industry and their related strength and weaknesses used to eliminate and minimize defects at requirements level. The stated RQ helps to explore defect identification and prevention techniques from academia and industry used to deal with the requirements defect types (see section 5.3).

To find related information in this regard, we proposed to accumulate qualitative data from academia and industry. The information for this RQ is achieved by dividing the RQ into two different but related sub-questions. The first sub-question (RQ3.1) deals with DPTs reported by literature research and industry (using interviews) while the other sub-question (RQ3.2) deals with the problems in these techniques which make them less efficient and less adoptable. The research in academia is concerned to find the information in second sub-question.

During industrial interviews the authors have noticed an interesting thing that companies even don't know about DPTs, defect identification and DP methods that are found in academic research. Some interviewees were curious about these DPTs and DP methods. Authors have provided enough aggregation of these methods and techniques (see section 5.3) according to their best knowledge. In this way software industry may get familiar with these requirements DPTs and DP methods to make the RE process more effective.

RQ3.1 - What is appropriate DPT for each defect types?

Appropriate techniques and methods used to deal with requirements defects are described in table 10 and 11 based on research from academia and industry simultaneously. Information in these tables is demonstrated on the basis of data analysis carried out explicitly from academia and empirical study results from section 5 and 6 respectively. Following table describes the techniques and methods reported by literature research along with the Company which follow these techniques. The detail of each technique can be uniquely found in section 5.3.

During interview it was very difficult to find DPT for a specific requirements defect type. Whenever authors asked questions to know DPT for each defect type, the interviewees could not tell them exactly. Since they were not using any DPT (see table 10) so how they can tell us what DPT handles which kinds of requirements defect types. However to identify defects in SRS, companies were using different method and techniques as given in table 11. You can also see in table 10 what types of defects are handled by which techniques those are practiced in six companies both in Pakistan and Sweden.

On the other hand the literature review results shows that DPTs like JAD, QFD, CRM, and participatory design do not focus on a particular defect type. They even don't only focus on requirements level defects but also do focus on design. They prevent defects in respective of defect type and SDLC phases.

To avoid rework in later stages, it is very important to prevent defects associated with requirements as early as possible. According to author's best knowledge they have given DPTs, DP methods, and defect identification techniques those can be used during the RE process to stop defects from penetrating them into later stages of SDLC. However the problem is unfamiliarity of companies with these DPTs and methods. The authors have tried to solve this problem by giving description of all DPTs, and DP methods in section 5.3.

It has been noticed in table 10 & 11 that most of the companies were using defect identification techniques instead of DPTs and DP methods. The comparison of these requirements analysis techniques proves that N-fold requirements analysis technique has been proved the best option because,

- Formal Inspection of SRS has been proved a good choice to identify defects that originate from the elicitation phase. It also achieves HDR (High Detection Rate) as compared to other requirements analysis techniques. HDR needs large numbers of inspectors to find defect from SRS. As we know large teams are known to be inefficient therefore instead of making large teams, N numbers of small teams were created in N-fold inspection. [78]. One more concern is that a fault that is not supposed to be found by single team (as in case of formal inspection) but it can be found if multiple teams (as in case of N-fold inspection) working on a single artifact [19]. It is also hoped that different inspection teams will find different defect or faults, in this way large number of teams will find more defects [39].
- It is true that prototyping removes gap between customer and developer but it can focus on only completeness and correctness of SRS. On the other hand N fold inspection has been proved versatile in finding defect types of defects with excellent rate of defect identification (discuss in percentages). During N fold inspection the analysts in multiple teams can identify defects that cannot be identified by a single team.
- It has been proved in an experiment that when N-fold inspection technique was used and appointed ten teams then defect detection rate raised up to 80% [19]. It means ten times more resources are required to catch 80% requirements level defects. But it is realty that most of the organizations believe in finding and fixing defects in later stages of SDLC [1] and don't want to appoint ten teams for the inspection of requirements. It means that N-fold requirement analysis technique is has HDR than prototyping and formal inspection but it needs more resources.

Table 10: Defect identification and DP techniques reported by literature research
Pakistani Companies = B, C, E
Swedish Companies = A, D, F

Sr. No.	Techniques Reported by Literature	Company
	Defect Identification Techniques	
1.	Formal Requirements Inspection	D, F
2.	Prototyping	A, B, C, E, F
3.	N-fold Inspection	
	Reading Techniques	
4.	Ad Hoc Based Reading	A, B, C, D, E
5.	Checklist Based Reading	A, D
6.	Scenario Based Reading (Defect Based Reading)	
7.	Perspective Based Reading	C
8.	Usage Based Reading	
9.	Function Point Reading	
10.	Metric Based Reading	
11.	Inspection Using Error Abstraction	
12.	Goal Oriented Requirements Analysis	
13.	Attributed GORA Technique	
	Defect Prevention Techniques	
14.	Formal Specification Method	
15.	Structural Analysis and Design Technique	
16.	Goal Based Requirements Analysis Method	
17.	Object Oriented Requirements Analysis	
18.	Joint Application Design (JAD)	
19.	Cleanroom Methodology (CRM)	
20.	Quality Function Deployment (QFD)	
21.	Participatory Design	

Following table 11 contains the defect identification and prevention techniques typically practiced in industry to deals with requirements defect types.

Table 11: Techniques practiced in industry

Pakistani Companies = B, C, E
Swedish Companies = A, D, F

Techniques	Description	Company
Mock-ups	This technique is used by organizations for creating user interface for understandability of functionality or features using paper or whiteboard for creation. Customer can participate and understand its purpose but using manual work takes too much time for complex interfaces.	A, C
Prototyping	Use prototyping of complex features to make requirements clear, understandable, unambiguous and to verify requirements correctness. Some organizations involve customer during this techniques to find missing and incomplete requirements.	A, B, C, E, F
Guidelines	It is found that some organizations used a pre-define set of instructions before starting the process. This technique is used to provide guidelines to requirements analyst to perform activities in process based on defined criteria to avoid problem in the process.	A, D
Informal meeting	An informal meeting is used to discuss problem in requirements. Some organization used check-list based to find problems in each requirement.	A
	Sometimes the requirements analyst arranged a session with developer and tester to make the requirements clear to them so that they all have a common understanding of requirements.	B
	Group meeting is also used where 4 to 5 people held a meeting to discuss problems in requirements. Stakeholders of this meeting are usually developers and operation manager.	C
Checklist based reading	The reading techniques is used in informal meeting or during inspection process to identify problems in requirements	A, D
Chat sessions	Some organizations establish open chat sessions with customers to make things understandable and clear if they found some ambiguities in requirements	B
Perspective-based reading	This technique is used during inspection to determine requirements from a particular perspectives i.e. customers, design, or implementation. During this approach misunderstanding and problems in requirements are discussed with the customers.	C
Test case generation	Some organizations used test-case based approach to determine the correctness of requirements. For this purpose tester is involved in the validation process which creates test cases for each requirement. If it is unable to create test cases for a requirement then it means there is something wrong with the specified requirements.	C
Ad-hoc	This technique is much more dependent upon the experience and knowledge of people involved in the RE process.	A, B, C, D, E,
Formal Inspection	To analyze requirements, inspection process is used to find problems in SRS with formal reviews.	D, F

Defect/trouble reports	To deal with customers reported and in-process defects, reported defects are documented in the form of a report. This report is later communicated with developer and analyst to learn about weaknesses and problem areas in the process.	D
	Problem log or log history is another mechanism to maintain and track software defects. The defects reported using this mechanism helps to identify incomplete and missing requirements as a result of malfunctioning or system crash.	E, C
Use cases	Use cases are used to manage and organize requirements for a system. It provides information about the interaction of users with the intended system from different perspective thus making requirements much more clear.	F

Some of the techniques described in literature in section 5.3, are not frequently practiced in industry like formal inspection, N-Fold inspection and DPTs. It is found that most of the organizations use ad hoc based reading techniques to find defects in SRS.

It is found that some organizations use prototyping of most complex features to make requirements clear and understandable. Prototyping and mock ups help in translating system requirements into executable form which help them to visualize and understand the system in a better way. This technique is very efficient to deal with some of the requirements problems especially those deals with critical user interface design. It is found that these techniques are costly and require time to implement, training to learn and resources. Initially at the start of project resources and cost are limited [79] so instead of making the prototype for the whole system, it would be a better approach to make it for most critical and most complex features.

Some organizations use guidelines before starting a process i.e. design, implementation. They also discuss problems and defects from the project of similar type from their defect or trouble reports. These techniques also help in minimizing defect if the organizations update their guidelines accordingly. It is found during empirical study that most of the organizations use guidelines for the design and implement purpose. They didn't care about the RE process and focus more on design and implementation. These guidelines are standards develop by the organization to guide designers and coders to develop things according to those standards. To make the RE process effective there must be some guidelines that contains thing that should be considered and taking care of before processing to the RE process. Chat sessions with customer to make things understandable are also helpful but they are not formal and can still leads to ambiguity, misunderstanding and omission due to remote communication and customer interest.

7.3.1 Comparison of SRS Reading Techniques

If you a look at table 10 then you will come to know that companies are using defect identification techniques instead of DP techniques or methods. It means at RE level they want to prevent defects just by finding and correcting them. Table 11 also shows that ad hoc and CBR techniques have been using frequently by the companies. If we have look at section 5.3 (reading technique) then we will come to know that there are better reading techniques than ad hoc and CBR techniques.

Porter and Votta in 1994 and Porter in 1995 have conducted multiple experiments in which they compared scenario based reading (SBR) with checklist and ad hoc reading techniques based on defect identification rate. The findings of experiments proved that scenario based or DBR has higher rate of defect identification than CBR and ad hoc based reading approach and checklist reading was no more effective than ad hoc. Practically it was proved that SBR or DBR captures 35% more defects than checklist and ad hoc approaches. [48]

Basili et al. have performed some experiments to know the effectiveness of PBR on SRS in NASA. He found that there is no major difference of reviewer's who were using PBR and those who were using other reading techniques like CBR but PBR reviewers performed well on generic documents. Laitenberger and DeBaud have also performed deep experiments to find effectiveness of PBR. They also found no significant difference in performance of reviewers when they were using PBR on code document. On the other hand Shull et al. said that PBR is better for reviewers who have some experience. According to above discussed of authors, it is proved that PBR reviewers can capture more defects as compared to the reviewer who use less systematic and less structured approach for review. They further added that PBR is more systematic, focused, goal oriented and tailor-able [39, 48].

In an experiment that was conducted by a group of students [49], they proved that UBR technique is more efficient in detecting faults than CBR and UBR is also efficient in finding different defects or faults than CBR (that is popular in industry). They proved that FPS technique is more efficient to inspect crucial areas of the software artifact (like SRS) than simple SBR technique. [50]

Authors had conducted an experiment to validate MBR technique. They found that it was more effective in the detection of defects than CBR technique. However it was not proved more efficient because it takes too much time to be implemented properly. [51]
It is proved from discussion above that DBR, PBR, UBR, and MBR are more effective than CBR and ABR techniques based on defect identification rate. It was noted that MBR is more effective than DBR but MBR takes too much time to be implemented properly. On the other hand FPR has been proved more effective than simple SBR technique. If we draw a sequence of SRS reading techniques with their effectiveness bases on defect identification rate then FPR will be on top and ABR technique will be at bottom. In other words FPR is more effective than SBR, CBR, and ABR techniques.

(GORA) technique provides a platform for traditional requirements analysis techniques (inspection, ad hoc based reading, prototyping etc) for their better performance and It also handle non functional requirements of the system. There is another requirement analysis technique (OORA) described in section 5.3.2.4 that is based on object oriented approach but OORA does not handle non functional requirements, and that way in [46] the author did move from OORA to GORA because GORA handles non functional requirements in the form of soft-goals. So GORA is better than OORA that provide reliable plate form for traditional requirements analysis techniques.

There is an enhance from of GORA technique that is called Attributed GORA technique because AGORA can help the requirements analyst to find inconsistency among the goals, to analyze the impact of requirements changes and to choose and adopt a goal from its alternatives. Requirements analyst can also judge the quality of SRS on the bases of its quality attributes such as correctness, unambiguousness, completeness and so on. [53].

RQ3.2 What are problems in existing DPT(s)?

This questions deals with identifying weaknesses in existing techniques in order to determine their adoptability and efficiency. The problems in existing techniques are described in detailed in section 5.3.

The preceding RQ illustrated the most commonly available DPTs described in literature research as well as those used in software industry. The purpose of those techniques is to facilitate the RE process to produce a quality SRS. They also facilitate the minimization of requirement related defects that cause major rework in later stages of SDLC. It is clear from table 10 that most of the techniques are not practiced in reality. Only few techniques described in literature (see section 5.3) are used in software industry. Here we can raise a question that why these techniques are not practiced by the industry? We can find answer of this question by taking empirical results into account because results shows those companies even did not hear about these DPTs (like A, B, C, D, E and F).

However if we do justice then we should say that a particular interviewee doesn't know about DPTs because we cannot generalize the things only on behave of single employee and there might be dozens or hundreds of employees with specific level of expertise and domain knowledge. For example when authors asked a question during interview with employee of company D in Sweden that "can we generalize things that you have told us"? They replied that you never do that because they have thousands of employees all over the world with different domains and departments. So the way of working might be different in different cities and countries. According to author's best literature review they found that DPTs have themselves some problems. The following tables 12 & 13 describe problems in DPTs based on section 5.3 and also explain which DPT and DP method have the best usage in particular conditions.

Table 12: Problems in defect identification techniques

Techniques Reported by Literature	Problems in Defect Identification Techniques	Best usage
Formal Requirements Inspection	• It needs trained people having good knowledge of inspection process and having good understanding of product [39]	Formal inspection has been proved versatile in finding different types of defects excellent defect identification rate.
Prototyping	• It needs about 10 percent budge of the whole project, it • It is very difficult to create a prototype of very large systems. • It needs trained peoples	Prototyping is used to take vital feedback from users to improve the quality of SRS. Its mean prototyping strongly depends on users feedback to develop a quality SRS.
N-fold Inspection	• Some faults were not found during inspection of SRS even by multiple teams, it means inspection process is good but not good enough. • Some faults of SRS cannot be discovered during inspection that needs execution in the form of design or implementation. These faults can be identified by using formal specification and design. • A flawed inspection team cannot perform well in the identification of faults • Multiple teams in N-fold inspection is costly	Prototyping becomes hard to implement for a complex and large features. It is true that prototyping removes gap between customer and developer but it can focus on only completeness and correctness of SRS. On the other hand N fold inspection has been proved versatile in finding defect types of defects with excellent rate of defect identification (discuss in percentages). During N fold inspection the analysts in multiple teams can indentify defects that cannot be identified by a single team.

Table 13: Problems in DPTs

Techniques Reported by Literature	Problems in DPT	Best usage
Formal Specification Method	• Formal method contains notations that are hard to understand for customers • Difficult to specify some aspects like requirements related to user interface • Software management feels hesitation to use a technique that is not widely used • Need extra ordinary expertise in formal systems and particularly in mathematical logics • Users feel hurdle in approving SRS because of difficult formal language	Formal methods are used where safety and security are critical
Structural Analysis and Design Technique	• Needs trained people • Complex in nature • Hard to implement • Used only to check problem elements but don't solve the problems	It does not solve the problem but it allows people to understand, manipulate or check problem elements.
Goal Based Requirements Analysis Method	• Difficult to draw hierarchy of goals • Complex in nature	It is very often in the organization that their goals remains unclear and are not provided easily. So to identify organization goals GOBRA is very significant to have as good information as possible to understand the domain, organization, process, and system [45]
Object Oriented Requirements Analysis	• Empirical study has proved that GORA technique can provide more detailed requirements definition than OORA techniques. • OORA method emphasis on static modeling and for real time system (such as distributed system) only OORA is not enough [67]. • OORA method does not handle non functional requirements, that's way in [46] the author did move from OORA to GORA because GORA handles non functional requirements in the form of soft-goals. • Needs good knowledge of object oriented methodology	The OOA techniques model the real-world environment that means an environment comprising of people, work processes, material things, and software systems [47]. In this way we can capture rich information that need to be model, analyzed, and understand before putting software system into implementation phase. It also provides better understanding of requirements specification and supports in OO design and implementation.

Joint Application Design (JAD)	• JAD needs training for users • Needs effective leadership	Resolve conflicts of varying stakeholder's needs toward systems in an open atmosphere by providing them opportunity to discuss and negotiate. Helps in eliminating potential defects related to requirements in the form of correctness, ambiguity, missing and incomplete requirements through continuous customer's involvement. [35]
Cleanroom Methodology (CRM)	• It needs trained peoples with good understanding of formal methods • Difficult to communicate SRS with customers • Complex in nature	CRM is used where safety and security are critical
Quality Function Deployment	• Difficult to apply on complex projects • Complex in nature	The main focus of QFD is to satisfy customer's needs

In addition to above discussion there could be some factors that can offer difficulty to adopt DPTs. These factors (extracted from section 5.3) are given below

- Complexity – some techniques are complex in nature and difficult to apply on projects. A typical example is the house of quality in QFD which is complex in nature and difficult to apply on complex project. Other example includes use of box structure in cleanroom methodology and hierarchy of goals in GORA techniques.
- Training and education – due to complex nature of techniques, they require particular training and education in order to apply them on projects e.g. use of formal methods limiting the authors of SRS to know about these method of specification.
- Domain specific – most of the techniques is domain specific and difficult to apply for all type of projects e.g. QFD for MIS system and prototype for user interface. It is therefore recommended to use more than one technique to attain certain goals.
- Resource consumption – some techniques are complex and take too much time to produce work products even for a single phase of SDLC. These techniques if applied on small projects can produce unrealistic results and leads to resource consumption. For example if the number of requirements increases then the size of house of quality matrix will increases and difficult to maintain. These techniques results in resource consumption in term of time, human and cost as well.

From the above discussion it is clear that most of the DPTs are obstacle for organizations due to the above problem factors. Other major problems for organizations are lack of awareness and research gap between industry and academia. They are not familiar with the DPTs and their significance as reported by literature research.

7.4 RQ4 & Data Analysis

RQ4 - How can we remove the problems in DPT and make them more efficient to handle defects that cause major rework?

DP is the process of finding the root causes of defects and prevents them from reoccurring [1]. However it is necessary to identify defect before finding their root causes and taking preventive actions. Authors have given detailed description of defect identification techniques like N-fold inspection, and prototyping in section 5.3.1. For requirements analysis, formal inspection has been proved an excellent technique [22]. During inspection process, inspectors need some sort of reading techniques for SRS. On the bases of literature review the authors have also given SRS reading techniques in section 5.3.1 that help in identifying requirements defect. This study mainly depends on early identification and prevention of requirements defects so that defects would not cause major rework down the development road. So defect identification in early stages gets importance and that's why the authors have given detail description of defect identification techniques with SRS reading techniques in section 5.3.1. In this way companies those are not familiar with DPTs, DP methods, defect identification techniques, and SRS reading techniques can have good enough introduction with preceding techniques.

On the bases of data analysis in section 7, the authors have given a list of recommendations and change in defect identification technique. N-fold inspection technique has been modified just to make it adoptable by those who don't use this efficient technique only due to its high cost and effort. There might be some questions in reader's mind that why we did select N-fold for modification? How modified N-fold would be cost effective and efficient? You can find answers of preceding questions in section 7.4.4

According to empirical study, most of the techniques are not widely used as shown in table 10 but you can see in the same table that most of the companies are using defect identification techniques to avoid requirements level defects from penetrating into later stages of SDLC. We know that the aim of this study is to avoid requirements level defects from penetrating down the software development road so that avoidable rework could be minimized (that is about 40 to 50 % of whole the project budget). So, on the bases of table 10 we can modify any defect identification technique to make it adoptable for organizations to get the best results regarding defect identification. On the bases of study findings the authors have provided solutions in three different areas that are given below.

- Improvement in a defect identification technique.
- Proposed defect taxonomy on the bases of figure 13 as defect preventive measure.
- Proposed a list of recommendations that will help in preventing defects at RE level.

7.4.1 Classification of Most Common Defect Types Using Boris Beizer Defect Classification Scheme

Based on defect identified from research in literature (see table 4) and through industry in the form of qualitative interview (see table 6 in section 7.1), the most common defects reported by both sources are summarized in table 8 and visually described in figure 12. It contains the information of most common requirements level defect types that originate from E and A&N phases of RE and cause major or minor rework in later stages of SDLC if they are identifies in later stages. In order to understand these defect types in more general form, there is need to classify them using some classification scheme to understand their class and details with respect to taxonomies discussed in section 5.2 in detail. To regulate these defect types into defect taxonomy, we selected Boris Bug Taxonomy (BBT) as guidelines for categorization of these defects. Following are the reasons for the selection of this specific taxonomy;

1) It is life cycle oriented taxonomy covering RE phase in detail.
2) Provides level of detail for defects (see appendix B) and also provide information about which defect belong to which type.
3) Provides definition for each defect type and associated defects as guidelines that make defects easy to adjust under a certain defect type.
4) Provides the flexibility to modify the taxonomy in case of an unadjustable defect type or defects that do not fit in the taxonomy.
5) According to available defects data (see table 8 and figure 12), this taxonomy is easy to use and implement.

The detail of BBT can be found in appendix B. Classifying the defects according to prescribed taxonomy surface new modifications due to some defects that are not described in the classification hence they are added under some defect types that make a consistent relation between defects and respective defect type. Most common defects found in table 8 and figure 12 falls into following major defect categories of BBT. [32]

1. Requirements Completeness – requirements as specified is either ambiguous and incomplete or overly specified.
2. Requirements Changes – requirements whether correct or not have been changed between the time programming started and testing ended.

7.4.1.1 Classification of Major Defect Types

The purpose of using BBT is to make a new classification of requirements defects types which is;

2. Simple and easy to understand.
3. Contains those defect types that are critical based on definition provided in section 7.1 under RQ 1.3 which describes the severity of defects in term of their rework in later stages of SDLC.

Although there are a number defects reported by research but it is clear from table 8 that some are major while other are minor. This information is also described in figure 12. By using BBT, a refined classification of major requirements level defects can be developed which will point out those defects that should be considered during RE process. The detail of classification with respect to defects summarized in table 8 in section 7.1 and figure 12 is given below [32].

1. **Undesirable requirements -** the requirement or a part of it is undesirable
    a. **Unwanted -** requirement is correct as stated but it is not desirable (Gold plating)
    b. **Unnecessary** - requirement is not needed. (Unnecessary requirements)
2. **Completeness** - the requirement as specified is either ambiguous, incomplete. or overly specified
    a. **Missing/unspecified -** the entire requirement is missing
        i. **Missing functionality -** the entire functionality or part of it is missing
        ii. **Missing features -** the feature customer wanted is missing
        iii. **Missing interface -** requirements related to interfaces or packages are missed
        iv. **Missing quality attributes -** (non-functional requirements): missing quality requirements
    b. **Omitted -** tacit or unwritten requirements
    c. **Unclearness -** lack of clarity in requirements
        i. **Ambiguity -** interpretation of requirements in more than one way thus creating different meanings of requirements.
        ii. **Misunderstood –** requirements wrongly understood
3. **Requirements volatility-** requirements, whether or not correct, have been changed between the time programming started and testing ended
    a. **Features -** requirement changes concerned with features
        i. **Feature added -** a new feature has been added
        ii. **Feature deleted -** previously required feature deleted
        iii. **Feature changed -** significant changes to feature, other than changes in cases
    b. **Cases -** cases within a feature have been changed. Feature itself is not significantly modified except for cases
        i. **Case added**
        ii. **Case deleted**
        iii. **Case changed -** processing or treatment of specific case(s) changed
    c. **Domain changes -** input data domain modified: e.g., boundary changes, closure, treatment
    d. **User messages and diagnostics -** changes in text, content, or conditions under which user prompts, warning, error messages, etc. are produced

e. **Internal interfaces -** direct internal interfaces such as call sequences, or indirect interfaces (e.g., via data structures) have been changed

f. **External interfaces -** external interfaces, such as device drivers, protocols, etc. have been changed

g. **Performance and timing -** changes to performance requirements (e.g., throughput) and/or timings

Classification of major defects based on table 8 from section 7.1 is described in more general form is described with the help of a diagram given below.

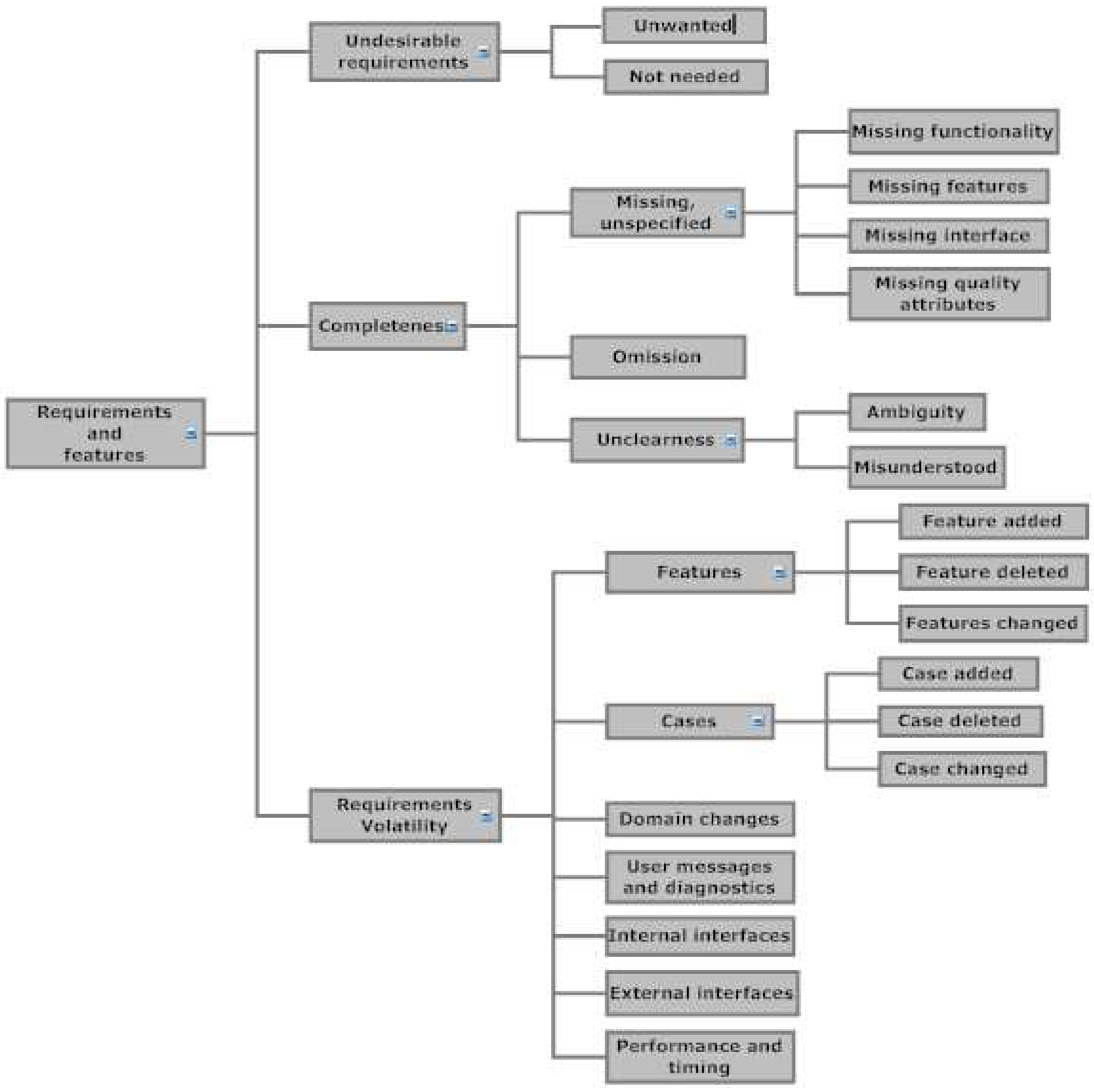


Figure 13: Classification of major defect types

The defects classification process start by mapping the defects types from summarized set of defect (see table 8 and figure 12) into above mentioned classification. This mapping helps to place a certain defect types under a specific bug category or sub-category is given below.

1) The first defect type is “ambiguity”. According to classification (figure 13), it comes under “Completeness” as this defect type deals with incomplete, ambiguous and overly specified requirements. Ambiguity is a sub-category having category type “Unclearness” deals with misinterpretation of requirements due to use of ambiguous words in SRS while another defect called misunderstanding deals with requirements which are wrongly misunderstood also place under the same defect type as shown in figure 13.

2) According to table 8, it is found that inconsistency is a minor defect (as reported by most of the companies) and resolved at requirements level so there is very little chance of it to escape into later phases and the rework for it is also minor so it is excluded from the taxonomy. Other defect that are minor in term of severity are also excluded from taxonomy
3) Defect type "Omission" means something that is tacit and not written down because it is elicited but it is in the mind of requirements analyst. It deals with completeness category of defect classification (see figure 13) under a new category.
4) "Unwanted" and "not needed" defects come under the first category which is "undesirable requirements". Once of the most common requirements risk is Gold platting which give extra features to customer although he does not demanded for it. This risk is the example of unwanted requirements. Which means requirements is written correctly but it is not required. God plating can increase project schedule and can put more effort on features that are not the actual demand of customers. The second defect "not needed" is demanded by customer although this feature is extra but it doesn't make any sense into the product except utilization of resources.
5) "Missing requirements" deal with functional as well as non functional requirements and contains missing features, missing functionality, missing interfaces and quality attributes etc. According to classification (see figure 13), it comes under "Completeness". We further divided it in term of missing functionality, feature, interface and quality attributes, so it must come under defect type "missing, unspecified" which means, the entire requirements or part of it is missed. A detail of sub-category that is added under "missing, unspecified" is given below

Table 14: Sub-categories and their definition added into classification

Type	Description	Example
Missing functionality	The entire functionality of part of it is missing	"User can save their name and password". An option will be given to the user to save the name and password is missing
Missing features	The feature customer wanted is missing	A mobile phone can provides video file feature with *.3GP and *.avi format. If it only support *.3GP format which means that the *.avi feature was missing
Missing interface	Requirements related to internal or external interface i.e. software and hardware interfaces respectively are missed	Software requirements such as third party interface is missing or Hardware requirements such as protocols, device drivers are missed
Missing quality attributes	Missing quality requirements	Missing user-interface requirements

6) Requirements volatility is the same as described by BBT because it's a major defect and contains complete detail in prescribed taxonomy in term of feature (added, deleted or modified), domain changes and interfaces changes etc.

Figure 13 shows detail of defect taxonomy in more simple form. It contains defects types and subsequent defects that can cause major rework in later stages of SDLC. The definition of each defect types can be found above. It is proposed to consider those defects that cause major lack in requirements. Although there are many other requirements level defects but their consequences are minor as found in figure 12. So it is better to put effort, resources and time for those defects which are major in term of their criticality, and severity.

7.4.2 List of Recommendations

On the bases of industrial and literature study findings (see chapter 7) the authors have given a list of recommendations that will help in preventing requirements level major defects. In this way rework can be minimized that has to be done when requirements defects are identified and fixed after implementation or product release.
This list of recommendation is given below in points.

1. In the beginning, the leader of RE team must held a start up meeting in which following information should be discussed
    a. Discuss and understand the importance of common requirements risks. The description of these risks can be found in section 5.1 and table 5 respectively. Common requirements risks are;
        i. Insufficient users environment
        ii. Creeping users requirements
        iii. Ambiguous requirements
        iv. Gold plating
        v. Minimal specification
        vi. Inaccurate planning
        vii. Traceability
        viii. Tacit requirements
    b. Discussion of bugs reported (requirements defects only) in last similar type of project.
2. Since the requirements elicitation is the most erroneous phase of RE as compared to other phases so people involved in this phase should have experience, training in term of technique(s) used in this phase and complete domain knowledge.
3. Requirements defect taxonomy should be used for defect classification that authors have given in section 7.4.2
4. Use IEEE SRS standards for writing requirements to make SRS unambiguous, complete, verifiable, consistent, modifiable, and traceable.
5. Follow proposed steps for requirements defect identification at RE level (see figure 17).
6. Industry need to learn about DPTs along with their strength and weaknesses in order to take advantage of them.
7. Industry should keep in contact with academia to open gateway for new research regarding DPTs and DP methods.
8. Most critical defects (described in figure 12) should be put on top priority during requirements analysis
9. More effort and time should be spent to identify and correct defect types discussed in taxonomy (see section 7.4.2) as compares to others.
10. Extraneous information, unstable requirements, and missing requirements should be identified and corrected at RE level to avoid project delay or failure.

7.4.3 Improvement in Defect Identification Technique

It was noted that companies were unfamiliar with DPTs were using defect identification techniques for requirements analysis. So there is needed to make defect identification more effective. The authors have found N-fold inspection is a good technique (however it needs more resources) to identify more percentage of requirements defects as compare to formal inspection and prototyping. It was noted during interviews that only two companies such as D and F were using formal inspection to find requirements defects but not even a single company was using N-fold inspection. The reason behind it could be its cost and extra resources requirements.

We cannot neglect the importance of an SRS because it plays pivotal role in the success or failure of the developing product [19]. It is bases for the rest of project planning, design, and coding. It is foundation for system testing and user documentation [4]. A complete and accurate description of product is very important before starting implementation [19]. A poor SRS can lead to a fail product. Defect correction in later stages of SDLC is more expensive, time consuming and hard to fix than in early stages [2, 3, 51]. It means early prevention of defects is very cost effective and could be helpful in minimizing rework in later stages of SDLC. We can prevent these defects from penetrating into later stages of SDLC by early identification process by an effective way.

Formal Inspection of SRS has been proved a good choice to identify defects that originate from the elicitation phase. It also achieves HDR (High Detection Rate) as compared to other requirements analysis techniques. HDR needs large numbers of inspectors to find defect from SRS. As we know large teams are known to be inefficient therefore instead of making large teams, N numbers of small teams were created in N-fold inspection. [78] We have described in chapter number 2 in detail that how N-fold inspection works. To identify maximum number of defects from SRS, we did converge our focus on N-fold inspection. It has many advantages that are given below

- The overlapping of fault detection by different team is minimal
- Early detection of faults in SRS becomes more effective
- Two or more teams inspecting same document can identify more faults as compared to single team.

The logic behind N-fold inspection is that a fault that is not supposed to be found by single team can be found if multiple teams are working on single artifact [19]. It is also hoped that different inspection teams will find different defect or faults, in this way large number of teams will find more defects [39].

Schneider, Johnny Martin and W.T. Tsai have conducted an experiment to identify requirements level defects by inspecting SRS. They noted that if only one team was assigned for the inspection of SRS then only 27% defects were caught. When they used N-fold inspection technique and appointed ten teams then defect detection rate raised up to 80% [19]. It means ten times more resources are required to catch 80% requirements level defects. However it is realty that most of the organizations believe in finding and fixing defects in later stages of SDLC [1] and don't want to appoint ten teams for the inspection of requirements.

Companies avoid N-fold inspection because it needs extra resources and time that makes it a costly process. To prevent requirements defect from penetrating into later stages of SDLC, it is very important to make N-fold inspection adoptable and efficient for all kinds of software development organizations. Here we feel that we should make an amendment in N-fold so that organizations can adopt it to find more defects within less cost. The authors thought over it that how it is possible if less number of teams may perform nearly equal to N-fold inspection?

We found during industrial interviews that companies were appointing not more than 15 to 18 people for single release or project. For example in company A there were three scrum teams having maximum 5 people in each team, and in company E there were also three teams having 5 to 8 people in each team. We also know that each inspection team requires 3 to 5 people. By keeping in mind the above figures, we decided to restrict N-fold Inspection up to four inspection teams with a moderator. This new modification in N-fold inspection is named as Four-fold inspection.

7.4.3.1 Four-fold Inspection

In four-fold inspection there are four inspection teams having 3 to 5 team members in each team. Single moderator is responsible for driving meeting in parallel and also responsible for handling results of all teams. Moderator (we recommend experienced requirements analyst or author) has direct contact with customer and team leader of each inspection team. In the first step these four teams are divided into two groups say group A and group B. The initial draft of SRS is also divided into two parts SRS-A and SRS-B based on requirements cohesiveness and requirements dependability. First part is provided to group A and second part to group B respectively and both groups A and B start inspection of their SRS parts. In the end of inspection process, moderator gathers effort of each group separately as shown in figure 14.

In step two before combining effort of both groups he distributes the SRS part of group A among the teams of group B and vice versa as shown in figure 15. The purpose of this distribution is to conduct informal peer review (walkthrough) of the SRS part of each group by teams of opposite group (both teams of group A will informally review SRS part of group B and teams of group B will informally review SRS part of group A). Walkthrough is an informal review technique as compare to formal inspection. It doesn't need defined procedure, no specified exit criteria, and meeting might be causal or disciplined [4].

It also does not need formal inspection steps like Planning, Overview, and Preparation. When both groups A and B complete inspection meeting they are not allowed to leave the room because they get second part of SRS that they haven't inspected yet (group A gets SRS-B and group B gets SRS-A) as shown in figure 16, then each team in each group start walkthrough meeting that is recommended to take time equal to or less than formal inspection meeting time.

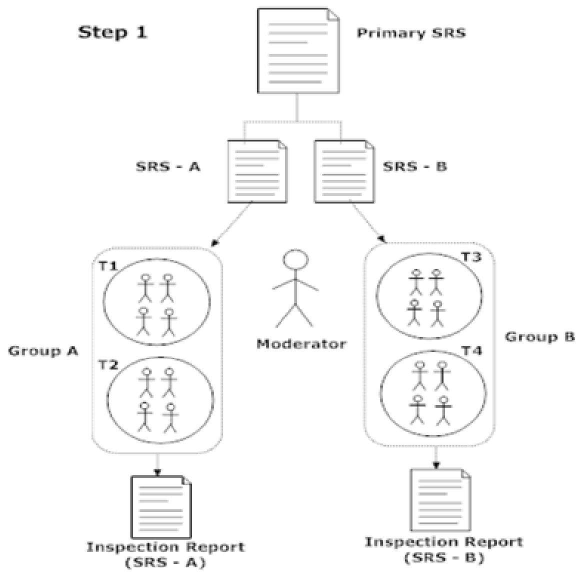
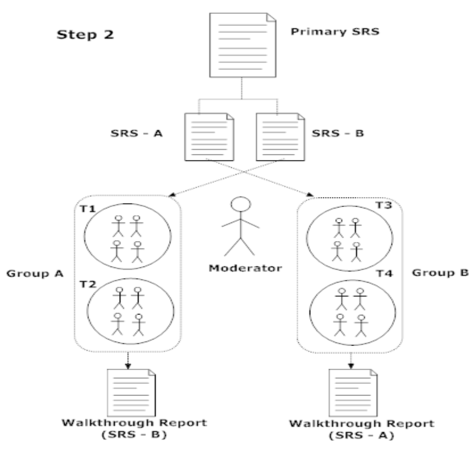


Figure 14: First step in Four-fold inspection

It means if formal inspection meeting takes time T then walkthrough meeting time (say W) can be represented as $W \leq T$. The author (requirements analyst) of SRS walks the reviewers through all requirements that he has written down. After the completion of informal peer review (walkthrough), step 3 gets start and moderator combines the effort of all teams and records detected faults in a database. Since there is probability of a fault or defect to be identified by multiple teams, so the moderator writes each fault once in the database.



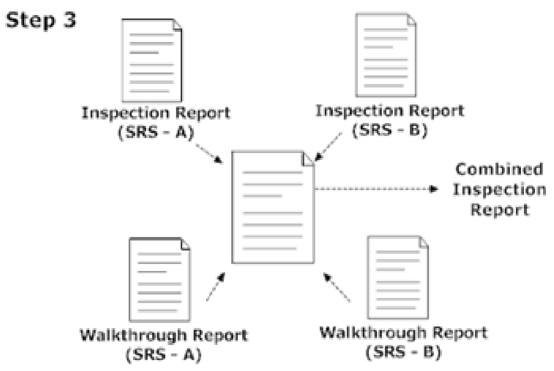

Step 3
Inspection Report (SRS - A)
Inspection Report (SRS - B)
Combined Inspection Report
Walkthrough Report (SRS - A)
Walkthrough Report (SRS - B)

Figure 15: Combined inspection report

Figure 16: Second step in Four-fold inspection

During walkthrough meeting there is no need to be prepared and it will be less formal than inspection meeting. All participants of walkthrough meeting would have background knowledge of the system that is going to be developed because they already have inspected one part of system's SRS. And team leader doesn't need to conduct planning, overview meeting, and preparation sessions because all members have got enough idea about system SRS by inspecting first part of SRS.

The purpose of including walkthrough peer review is to save time and resources that are needed in N-fold inspection process for second part of SRS. We divided initial draft of requirements into two parts for N-fold inspection because each inspection team will spend half time as compare to total time of inspection process. We can save second half time of inspection process but we have to spend time on walkthrough meeting (equal to or less than formal inspection meeting). It means walkthrough can save time and resources that supposed to be spent during formal inspection process in the form of Planning, overview meeting, and preparation effort of second half of SRS.

As we know that inspection process requires a specific reading technique like CBR, PBR and SBR. These techniques help the inspector for better identification of requirements defects. For four-fold inspection the authors have recommended scenarios based reading (defect based reading) technique because it is more efficient than ad hoc based reading and CBR technique [48, 39]. It can capture more than 35% more defects than ad hoc and CBR [48]. In addition SBR provides good opportunity to focus on defect taxonomy to prevent requirements defects. As the authors have developed their own requirements defect taxonomy (see figure 13) based on BBT and based on major defect types described in figure 12.

We have summarized in figure 13 the most common and major requirements level defects that can cause major rework if they would be identified and fixed after implementation. So it is very important to identify them during the RE process. As the authors have put all these major defects in the taxonomy that they have developed, so this taxonomy must be focused by an efficient reading technique like SBR. We hope that combination of our defect taxonomy and SBR in Four-fold inspection would be best combination to identify and prevent requirements level defects. For better results FPBR can be used instead of SBR.

By using SBR, inspection is done based on different scenarios that are defined according to defect taxonomy in use [39]. Defects are categorized into different classes and for each class of defect a set of questions are developed. Scenarios are also considered to focus a specific view point that helps to identify particular type of defects. For example say a company is developing Railway Track Crossing system. During inspection of SRS for this project, reviewer can make a scenario like "when train reaches 5Km away from the railway track barrier then barrier will start to come down. Let if barrier stops in the middle and do not come down completely then what could be problems".

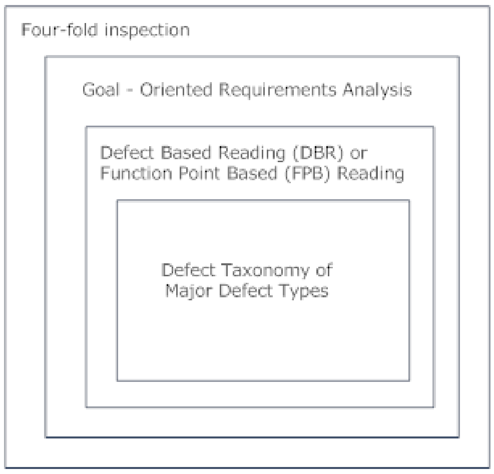

Four-fold inspection
Goal - Oriented Requirements Analysis
Defect Based Reading (DBR) or Function Point Based (FPB) Reading
Defect Taxonomy of Major Defect Types

Figure 17: Steps in requirements defect identification

It is very often in the organization that their goals remains unclear and are not provided easily. So to identify organization goals it is very significant to have as good information as possible to understand the domain, organization, process, and system [45]. Since the GORA technique provides a platform for traditional requirements analysis techniques (inspection, ad hoc based reading, prototyping etc) for their better performance and it makes organizational goals clear. It also handles non functional requirements of the system in the form of soft-goals and is better than OORA method. So the SRS review process (that is done by using SBR and defect taxonomy created by the authors) should be based on GORA technique. The overall process of defect identification recommended by the authors would adopt a shape as described in figure 17. Besides that, employees should have completed understanding of scenario based reading technique and GORA technique in order to implement it efficiently. The authors have given sufficient detail about these techniques in sections 5.3.1.5 and 5.3.2.3

7.5 RQ5 & Data Analysis

To what extent the prevention actions are valid?

7.5.1 Validation of Study Finding

In this section we will discuss about the validation of our finding from research study. The validation is based on designed described in section 4.5. Based on study finding, authors have designed questionnaires given below in table 15. This questionnaire will be used during interview meeting.

Table 15: Questionnaires for study finding

S.No.	Questions
1	What do you think about start-up meeting and the things that should be discussed in the meeting? Would it be proved effective in preventing requirements level defects?

2	Should it be (start-up meeting) beneficial for improving overall RE process and minimizing requirements defects?
5	Do you think the use of maximum number of scenarios for requirements help to overcome missing and incompleteness in requirements?
6	Do you think that use of IEEE standards for specifying requirements helps to improve SRS quality?
8	Do you think that the information obtained from classification of defects proves to be valuable in start-up meeting?
9	Do you think training, experience and domain knowledge can improve elicitation process?
10	Do you think an SRS that is unambiguous, complete, verifiable, consistent, modifiable, and traceable is a quality SRS?
11	Do you think Four-fold inspection is an adoptable requirements analysis technique as compared to N-fold inspection?
12	Do you think that defect detection rate of Four-fold inspection would be low, medium or high?
13	Do you think Four-fold inspection would catch requirements defects nearly equal to N-fold inspection (where N=4)?
14	Do you think Four-fold inspection is a cost effective inspection technique as compared to N-fold inspection?
15	Do you think the combination of our proposed requirements defect taxonomy and scenario based reading (defect based reading) technique proposed in Four-fold inspection can identify good number of major requirements defects?
16	What do you think about our proposed solution (i.e. list of recommendations, Four-fold inspection technique, and requirements defect taxonomy) would prove a good attempt to prevent requirements level defects and to minimize rework caused by those defects in later stages of SDLC?

7.5.2 Validation Execution

7.5.2.1 Validation Feedback from Company C

In order to validate the study finding, the interviewee from Company C is selected for this purpose. The detail of Company and interviewee can be found in section 6.1.3. Before going directly to the questionnaires described in table 14, a short presentation is given to the interviewee as a background of thesis. The answers of interviewee with respect to questionnaires are described in appendix C.

According to interviewee, the start-up meeting can play an important role in improving the RE process and minimizing defects. The use of defects reports of previous similar projects and risk that might affect the performance of the RE process in a positive way. According to interviewee, start-up meeting will consume time but if organization can plan things that are mentioned in start-up meeting in advance then they can avoid them before they get serious. Besides that people involved in the requirements elicitation process should be experience and have training if required to make this process much more effective. For Incompleteness and missing requirements, use of scenario-based approach helps to overcome them at requirements level. The interviewee also agrees with the use of IEEE standards for improve the quality of SRS. This will help to overcome a lot of problem arise from poorly written documents.

According to interviewee, four-fold inspection techniques is a better approach to overcome the major requirements defects as compare to N fold inspection where N = 4 and it depends upon the size of document in term of number of requirements. Interviewee proposed that this technique could be efficient for projects having 200 requirements but in case of requirements greater than 400 four-fold could be limited. It takes equal resources as compare to N fold but the cost and time are less consumed. Based on interviewee experience, four-fold inspection will catch 42% defects than normal N fold inspection for N = 4. The use of scenario based approach in four-fold make this techniques efficient because according to interviewee while inspecting SRS using scenario-based reading, one can see requirements from actual users perspective with respect to their actual use in users domain. Thus this approach helps in understand and finding lack in requirements more. According to interviewee, it is a better way to overcome requirements level defects to avoid there ripple effect in later phases of SDLC.

7.5.2.2 Validation Feedback from Company D

The second interviewee from Company D was selected for validation. The detail of Company and interviewee can be found in section 6.1.2. The interviewee fully agrees with the start-up meeting for the improvement of the RE process and minimizing requirements defects. According to interviewee, tacit requirements are the primary risk to the RE because both analyst and customer got tacit requirements on their own. It is better if you discuss them in advance to provide an awareness to consider it during the RE process.

Interviewee also agrees with the concept of prior requirements elicitation training, analyst experience and sufficient domain knowledge about proposed system. To uncover requirements defects like incompleteness and missing, scenarios could prove to be a better approach. To overcome SRS related problems, use of standard template for specifying requirements will also improve this process. The interviewee also aggress with the concept of short meeting after testing and maintenance phases and use of taxonomy for classification of defects would be a good approach to classify and learn about defects. A part from that, interviewee agreed that four-fold inspection is adoptable and can work efficiently as compare to N-Fold for four teams. According to interviewee, this technique is cost effective, identify defects equal to N-fold inspection (for N=4) and it would be a good approach to use scenario based reading and GORA techniques with four-fold inspection to get maximum benefits. However, in order to implement it, employees must have experience and training of techniques used in four-fold inspection as proposed by authors.

7.5.3 Lesson Learnt

Based on validation of study finding from two interviewees, we infer the following points.

- Risks related to RE that might affect the process or create other requirements related problems should be considered so that they can be minimized.
- Use of formal RE process with defined roles and responsibility along with training (if required) could also help to overcome lack in requirements.
- It is a beneficial if in-process and operation reported defects related to requirements are properly classified
- Use of start-up meeting would be valuable for overall improvement of the RE process and minimization of requirements related defects.
- Use of scenarios helps in identifying missing and incomplete requirements
- To avoid SRS related issues, use of standards like IEEE will be favorable.
- Four-fold inspection with scenario based reading would be a better approach during requirements analysis to uncover maximum number of defects.

.

8 RQS &ANSWERS TO RQS

This section contains the summary of RQs and answers to each RQ.

8.1 RQ1 (RQ1.1, QR1.2, RQ1.3)

The most common requirements level defect types reported by research literature based on E and A&N are missing, omissions, ambiguous, and volatile requirements. There are some risks that are associated with the RE process. These risks can play unpleasant role in the origination of defects associated with the requirements. In table 5, it has been mentioned that which requirements risk can cause what defect type and what kinds of consequences it has. Most of the risks are related to elicitation phase of RE that directly related to customer or users. During industrial interviews the authors have noticed that most of companies are not taking care of these risks and merely depending on their experience and knowledge.

Each defect types can cause rework down the development road in the form of "Major" or "Minor" rework. This has been described in table 6 against each defect types under column "Severity Level". The severity level (major or minor) of defect type describes the rework caused by each defect if it would be identified and fixed in later stages of SDLC (design, implementation, testing, and maintenance).

The volatile requirements, ambiguity, missing quality attributes, misunderstood, omission, unwanted/unnecessary, missing requirements are requirements defect types with high severity level (major) and mutually reported by industry research and literature study. The reasons for each defect types are given in table 8 under the column third (named defect reasons). The defect types having high severity and high rate of occurrence is only missing requirement and it is reported by all six companies from Pakistan and Sweden. All major defect types do not have high rate of occurrence and all minor defect types don't have low rate of occurrence (see figure 12).

8.2 RQ2

Defect types reported by Pakistani companies are more in number as compared to Swedish companies. Defects are mostly reported by customers and according to interviewee sometime they are greater than 60% (company B). They also reported omissions, extraneous information, inconsistencies, conflicts (rare), misunderstanding (20 to 30 % reported by company E) and ambiguous requirements (5% to 30% reported by company E). According to Pakistani company B, more than 74% (this percentage is about defect finding only in RE phase) defects are identified and fixed during RE process. Defects reported during testing are estimated to 60% to 70% (B & C) and rest of the defects is reported by customers after release which is more than 60 % (company B). Swedish Companies like D and F face less defect types in development and low defect rates after releasing their products. The organization (D) uses extensive testing in order to deal with defects before release. The companies from both countries freeze SRS before starting implementation.

There is only one company (B) in which most of the defects are reported by the customer and in rest of companies mostly defects are reported by the software tester. Most of the companies (A, B, E, and F) don't use IEEE for SRS development and believe in finding requirements defects in early stages but they don't have any formal DPT (see table 10) for early defect prevention and they do not categorize their defects according to defect taxonomies given in section 5.2. Swedish companies have more formal RE process than Pakistani companies. One very important practice that has been neglected by most of companies (both Pakistani and Swedish) is root cause analysis.

It has been noted that interviewees from both countries were not aware of DPTs or DP methods that have been shown in table 10 and instead they were using defect identification techniques as shown in table 11. The maximum numbers of software developers for a single project (project module or release) were reported not more than 20 by some Pakistani and Swedish companies.

8.3 RQ3 (RQ3.1, RQ3.2)

During interview it was very difficult to find DPT for a specific requirements defect type. Whenever the authors asked questions to know DPT for each defect type, the interviewees could not tell them exactly. Since they were not using any DPT (see table 10) so how they can tell us what DPT handles which kinds of requirements defect types. However to identify defects in SRS, companies were using different method and techniques (defect identification techniques) as given in table 11. You can also see in table 11 what types of defects are handled by which techniques those are practiced in six companies both in Pakistan and Sweden. Since companies are not familiar with DPTs and DP methods, so according to author's best knowledge they have given DPTs, DP methods, and defect identification techniques those can be used during requirements analysis to stop defects from penetrating them into later stages of SDLC.

The comparison of formal requirements inspection technique, N-fold inspection technique, and prototyping as requirements analysis technique proves that N-fold inspection has high defect identification rate but it needs more resources. The comparisons of SRS reading techniques (given in section 5.3) show that SBR identifies more defects than ABR and CBR techniques, and FPR technique identifies more defects than SBR or DBR techniques. On the other hand GORA technique is more efficient than OORA technique for requirements analysis.

It has been revealed in tables 12 & 13 that which DPT or DP method has what kinds of problems and under third column ("Best Usage") you can see the best usage of each DPT or DP method.

8.4 RQ4

The description of RQ4 explains that after finding problems in existing DPTs practicing in software industries, how we will remove these problems? It might be change in existing DPT or creation of new DPT or we can give a list of recommendations. In response of this question the authors have modified N-fold requirements analysis technique on the bases of study findings and proposed to use SBR, GORA techniques, and defect taxonomy (developed by the authors on the bases of study finding) within modified N-fold inspection (four-fold inspection, see sections, 7.4.1.1, 7.4.3 and figure 17). The authors have also proposed a list of recommendations to avoid requirements level defects from going down the development road.

8.5 RQ5

The validation is based on design described in section 4.5. Based on study finding, authors have designed questionnaires given in table 15. This questionnaire has been used during interview meeting. The interviews were conducted from two companies and the interviewees had good enough background knowledge of all proposed solutions that has been discussed in RQ4.

9 CONCLUSIONS

Defect prevention (DP) in early stages of software development life cycle (SDLC) is very cost effective than in later stages. The requirements elicitation and analysis & negotiation (E and A&N) phases in requirements engineering (RE) process are very critical and are major source of requirements defects. A poor E and A&N process may lead to a software requirements specifications (SRS) full of defects like missing, ambiguous, inconsistent, misunderstood, and incomplete requirements. If these defects are identified and fixed in later stages of SDLC then they could cause major rework by spending extra cost and effort. Organizations are spending about half of their total project budget on avoidable rework and majority of defects originate from RE activities.

The aim of this thesis to reduce the rework in later stages of SDLC that is caused by requirements defects originating from E and A & N phase in the BESRE. To fulfill the purpose of this study, qualitative research approach has been adopted (empirical and literature studies are presented in this thesis). The empirical study is carried out with the help of six companies from Pakistan & Sweden by conducting interviews while literature study is done by using literature reviews.

Most commonly reported defect types by literature study are missing requirements, omissions, ambiguous, and volatile requirements. The major defect types reported by industry research are missing requirements, unnecessary/unwanted requirements, volatile requirements, misunderstood and unstable requirements. On the other hand volatile requirements, ambiguity, missing quality attributes, misunderstood, omission, unwanted/unnecessary, missing requirements are requirements defect types with high severity level and mutually reported by industry research and literature study (see figure 12). To fulfill the purpose of this study, all major defects having high rate of occurrence have been put on top priority and is recommended to spend more time and effort to prevent them from reoccurring during RE process and which can ensure reduction in rework down the development road. The reasons of these defects are described in section 7.1. It is also found that defects are reported by tester most of the time (see table 9).

The defect types having high severity and high rate of occurrence is only missing requirement and it is reported by all six companies from Pakistan and Sweden. Some major defect types have medium rate of occurrence like omissions, misunderstood requirements, unwanted/unnecessary, and missing quality attributes. On the other hand some defect types have low rate of occurrence but severity level is high such as ambiguity, and volatile requirements. The defect types having high rate of occurrence and minor severity level are inconsistency, and misunderstood requirements, extraneous information has low rate of occurrence and minor severity level, and unclearness has high rate of occurrence but minor severity level with respect to rework in later stages of SDLC (see figure 12)

Extraneous information, unstable requirements and missing requirements can affect the project plan in the form of project delay or failure.

Swedish companies have more formal RE process as compared to Pakistani companies. Most of the companies do not use IEEE format for SRS development and freeze SRS before starting implementation that means they have low requirements volatility rate. It was found during industrial study that not more than 20 software developers were appointed for a single release, project, or module.

It is also found that both Pakistani and Swedish companies believe in DP in early stages but they are not using any formal DPT or DP method found in literature. They are also not familiar with defect taxonomies (see section 5.2), and current research about DPTs and DP (see section 5.3) methods instead they are using and relying on defect identification techniques. Apart from unfamiliarity of industry from DTS, It is observed from literature research that most of the DPTs are complex in nature and require training in order to implement them.

N-Fold inspection technique finds more defects than informal inspection and prototyping during software requirements analysis. On the other hand SBR technique identify more defects than CBR and ABR techniques however FPB reading technique is more effective than SBR with respect to defect identification rate. It is also noted that GORA technique is more efficient than OORA to conduct good quality requirements analysis.

On the bases of study findings the authors have proposed solutions in three different areas that are 1) Steps in requirements defect identification, where N-fold requirements inspection technique has been modified in the form of four-fold requirements inspection. It is proposed to use SBR or FPBR technique as SRS reading technique during inspection process by using defect taxonomy developed by the authors that purely covers major defect types described in figure 12. It is also proposed that the SRS review process (that is done by using SBR and defect taxonomy created by the authors) should be based on GORA technique (see figure 17), 2) Proposed defect taxonomy on the bases of major defect types mutually reported by industry and literature research (see figure 12) as defect preventive measure, and 3) Proposed a list of recommendations that will help in preventing defects at RE level. The authors hope that preceding measures will help in preventing requirements level defects at RE level and there will be reduction in avoidable rework in later stages of SDLC.

10 FUTURE WORK

To confirm the efficiency and benefits of proposed taxonomy, it will be beneficial to apply it on a defect report of a real project. This will help to understand the importance of classification of defects and to learn about the types of defects. To determine the effectiveness of proposed steps to indentify requirements level defects, it is necessary to perform an experiment in a controlled environment. This will help in making decision about whether four-fold works better in the presence of GORA, SBR, and defect taxonomy proposed by the authors are efficient to identify requirements defects as compare to traditional N-fold requirements analysis.

Different companies have different types of projects with different domains, different software development methodologies, different bases (product based or project based), and different software life cycles. So it would be interesting if case study would be conducted for particular company to prevent requirements defects and to avoid rework in later stages, because case study of a company provides detailed information for a specific domain.

After industrial interviews authors found that there are lots of DPTs and DP methods like QFD, CRM, and JAD from which companies are not familiar. It would be interesting if someone try to make them adoptable by convincing the companies with their advantages and importance.

11 REFERENCES

[1] Mays, R. G., Jones, C. L., Holloway, G. J., and Studinski, D. P. 1990. Experiences with defect prevention. *IBM Syst. J.* 29, 1 (Jan. 1990), 4-32.

[2] Kosman, R. J. 1997. A two-step methodology to reduce requirement defects. *Ann. Softw. Eng.* 3 (Jan. 1997), 477-494.

[3] Boehm, B. and Basili, V. R. 2001. Software Defect Reduction Top 10 List. *Computer* 34, 1 (Jan. 2001), 135-137. DOI= http://dx.doi.org/10.1109/2.962984

[4] Wiegers, K. E. 2003 *Software Requirements*. 2nd Edition. Microsoft Press.

[5] Peled, A.; Salzman, L; Danon, A.; Rogoway, P.; Defect Prevention Techniques for High Quality and Reduced Cycle Time, An ESSI Process Improvement Experiment (PIE), Motorola Communication Israel Ltd. DOI = http://www.iscn.at/select_newspaper/measurement/motorola2.html

[6] Prasad, R. Roger, L. Thomas, A. Chia-Chu, C. and Dale, K. 2005 A new approach for software requirements elicitation, Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, and First CIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on 23-25 May 2005 Page(s):32 - 42 DOI = 10.1109/SNPD-SAWN.2005.5

[7] Kotonya, G. and Sommerville, I., Requirements Engineering Process and Techniques. In: pp. 294

[8] Tian, J. 2005. Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement, Willey-IEEE Computer Society Press, ISBN: 9780471713456

[9] McDonald, M., Musson, R., and Smith, R. 2007 The Practical Guide to Defect Prevention. First. Microsoft Press.

[10] Langari, Z. and Pidduck, A. B. 2005. Quality, cleanroom and formal methods. In Proceedings of the Third Workshop on Software Quality (St. Louis, Missouri, May 17 - 17, 2005). 3-WoSQ. ACM, New York, NY, 1-5. DOI= http://doi.acm.org/10.1145/1083292.1083302

[11] Kashif, A. Ahmad, S. and Akhtar, S. 2005. Defect Prevention Techniques and its Usage in Requirements Gathering - Industry Practices S. National University of Computer and Emerging Sciences. Student Conference on 27-27 Aug. 2005 Page(s): 1-5

[12] Lauesen, S.; Vinter, O.; 2000 Preventing Requirements Defects: An Experiment in Process Improvement, Requirements Engineering Journal, In Proceedings of the Sixth International Workshop on Requirements 2000, pp. 37-50.

[13] Hazzan, O., Dubinsky, Y., Eidelman, L., Sakhnini, V., and Teif, M. 2006. Qualitative research in computer science education. In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (Houston, Texas, USA, March 03 - 05, 2006). SIGCSE '06. ACM, New York, NY, 408-412. DOI= http://doi.acm.org/10.1145/1121341.1121469

[14] Levy, Y. and Ellis, T.J. 2006. A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research. Informing Science Journal Vol. 9 No. 1 pp. 181-212

[15] Hancock, B 1998. An Introduction to Qualitative Research. UK: Trentfocus Org.

[16] Creswell, J. W. (2003). Research design: Qualitative, quantitative, and mixed methods approach. Thousand Oaks, CA: Sage Publications, Inc.

[17] Pressman, Roger S. (2005). Software Engineering: A Practitioner's Approach, 6th Edition. McGraw-Hill. ISBN: 978-0-0728-5318-6.

[18] Gorschek, T., 2006. Requirements Engineering Supporting Technical Product Management. Ph.D. Thesis, Department of Systems and Software Engineering, Blekinge Institute of Technology, Ronneby, Sweden

[19] Martin, J. and Tsai, W. T. 1990. N-Fold inspection: a requirements analysis technique. *Commun. ACM* 33, 2 (Feb. 1990), 225-232. DOI= http://doi.acm.org/10.1145/75577.

[20] Schneider, G. M., Martin, J., and Tsai, W. T. 1992. An experimental study of fault detection in user requirements documents.*ACM Trans. Softw. Eng. Methodol.* 1, 2 (Apr. 1992), 188-204. DOI= http://doi.acm.org/10.1145/128894.128897

[21] Walia, G. and Carver, J. "A Systematic Literature Review to Identify and Classify Software Requirements Errors. Journal of Information and Software Technology.

[22] Shull, F., Rus, I., and Basili, V. 2000. How Perspective-Based Reading Can Improve Requirements Inspections. *Computer* 33, 7 (Jul. 2000), 73-79. DOI= http://dx.doi.org/10.1109/2.869376

[23] Jones, C., 1997. Software quality analysis and guidelines for success, International Thomson Computer Press, pp. 158 – 164

[24] Hayes, J. H. 2003. Building a Requirement Fault Taxonomy: Experiences from a NASA Verification and Validation Research Project. In *Proceedings of the 14th international Symposium on Software Reliability Engineering* (November 17 - 21, 2003). ISSRE. IEEE Computer Society, Washington, DC, 49.

[25] Lauesen, S. and O. Vinter 2001. Preventing requirement defects: An experiment in process improvement, *Requirements Engineering Journal* 6(1): 37-50.

[26] Seaman, C. B., Shull, F., Regardie, M., Elbert, D., Feldmann, R. L., Guo, Y., and Godfrey, S. 2008. Defect categorization: making use of a decade of widely varying historical data. In *Proceedings of the Second ACM-IEEE international Symposium on Empirical Software Engineering and Measurement* (Kaiserslautern, Germany, October 09 - 10, 2008). ESEM '08. ACM, New York, NY, 149-157. DOI= http://doi.acm.org/10.1145/1414004.1414030

[27] Zowghi, D. and Nurmuliani, N. 2002. A Study of the Impact of Requirements Volatility on Software Project Performance. In*Proceedings of the Ninth Asia-Pacific Software Engineering Conference* (December 04 - 06, 2002). APSEC. IEEE Computer Society, Washington, DC, 3.

[28] Malaiya, Y. K. and Denton, J. 1999. Requirements Volatility and Defect Density. In *Proceedings of the 10th international Symposium on Software Reliability Engineering* (November 01 - 04, 1999). ISSRE. IEEE Computer Society, Washington, DC, 285.

[29] Stark, G. E., Oman, P., Skillicorn, A., and Ameele, A. 1999. An examination of the effects of requirements changes on software maintenance releases. *Journal of Software Maintenance* 11, 5 (Sep. 1999), 293-309. DOI= http://dx.doi.org/10.1002/(SICI)1096-908X(199909/10)11:5<293::AID-SMR198>3.0.CO;2-R.

[30] Van Lamsweerde, A. 2000. Requirements engineering in the year 00: a research perspective. In *Proceedings of the 22nd international Conference on Software Engineering* (Limerick, Ireland, June 04 - 11, 2000). ICSE '00. ACM, New York, NY, 5-19. DOI= http://doi.acm.org/10.1145/337180.337184

[31] Vinter, O., Lauesen, S., and Pries-Heje, J. 1998. *A Methodology for Preventing Requirements Issues from Becoming Defects (PRIDE)*. ESSI Project No. 21167. Final Report. http://www. esi.es/ESSI/AII/21167

[32] Beizer, B. 1990 *Software Testing Techniques (2nd Ed.)*. Van Nostrand Reinhold Co.

[33] McDonald, M., Musson, R., and Smith, R. 2007 *The Practical Guide to Defect Prevention*. First. Microsoft Press.

[34] Vinter, O., Lauesen, S. 2000. Analyzing Requirements Bugs, Software Testing & Quality Engineering, Volume 2.

[35] Chin, K. F. 1995. A JAD experience (abstract). In *Proceedings of the 1995 ACM SIGCPR Conference on Supporting Teams, Groups, and Learning inside and Outside*

the IS Function Reinventing IS (Nashville, Tennessee, United States, April 06 - 08, 1995). L. Olfman, Ed. SIGCPR '95. ACM, New York, NY, 235-236. DOI= http://doi.acm.org/10.1145/212490.213690

[36] IEEE Software Staff 1993. Defect-Causal Analysis Drives Down Error Rates. *IEEE Softw.* 10, 4 (Jul. 1993), 98-99. DOI= http://dx.doi.org/10.1109/52.219639

[37] Javed, T., Maqsood, M. e., and Durrani, Q. S. 2004. A study to investigate the impact of requirements instability on software defects. SIGSOFT Softw. Eng. Notes 29, 3 (May. 2004), 1-7. DOI= http://doi.acm.org/10.1145/986710.986727

[38] Zowghi, D. and Nurmuliani, N. 2006. Requirements Volatility and Its Impact on Change Effort: Evidence-based Research in Software Development Projects. AWRE 2006 Adelaide, Australia

[39] Aurum, A., Petersson, H. and Wohlin, C., "State-of-the-Art:Software Inspections after 25 Years", Software Testing, Verification and Reliability, 12(3):133-154, 2002.

[40] Martin, J. and Tsai, W. T. 1990. N-Fold inspection: a requirements analysis technique. Commun. ACM 33, 2 (Feb. 1990), 225-232. DOI= http://doi.acm.org/10.1145/75577.75587

[41] Lamsweerde, A. v. 2000. Formal specification: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (Limerick, Ireland, June 04 - 11, 2000). ICSE '00. ACM, New York, NY, 147-159. DOI= http://doi.acm.org/10.1145/336512.336546

[42] Gomaa, H. and Scott, D. B. 1981. Prototyping as a tool in the specification of user requirements. In Proceedings of the 5th international Conference on Software Engineering (San Diego, California, United States, March 09 - 12, 1981). International Conference on Software Engineering. IEEE Press, Piscataway, NJ, 333-342.

[43] Ross, D. T. and Schoman, K. E. 1979. Structured analysis for requirements definition. In Classics in Software Engineering, E. N. Yourdon, Ed. ACM Classic Books Series. Yourdon Press, Upper Saddle River, NJ, 363-386.

[44] Ross, D. T. 1985. Applications and Extensions of SADT. Computer 18, 4 (Apr. 1985), 25-34. DOI= http://dx.doi.org/10.1109/MC.1985.1662862

[45] Anton, A. I. 1996. Goal-Based Requirements Analysis. In Proceedings of the 2nd international Conference on Requirements Engineering (ICRE '96) (April 15 - 18, 1996). ICRE. IEEE Computer Society, Washington, DC, 136.

[46] Mylopoulos, J., Chung, L., Liao, S., Wang, H., and Yu, E. 2001. Exploring Alternatives During Requirements Analysis. IEEE Softw. 18, 1 (Jan. 2001), 92-96. DOI= http://dx.doi.org/10.1109/52.903174

[47] Mylopoulos, J., Chung, L., and Yu, E. 1999. From object-oriented to goal-oriented requirements analysis. Commun. ACM42, 1 (Jan. 1999), 31-37. DOI= http://doi.acm.org/10.1145/291469.293165

[48] V. Basili, G. Caldiera, F. Lanubile, and F. Shull, "Studies on Reading Techniques," Proceedings of the Software Engineering Workshop, Greenbelt, MD, pp. 59-65, December 1996.

[49] Thelin, T., Andersson, C., Runeson, P., and Dzamashvili-Fogelstrom, N. 2004. A Replicated Experiment of Usage-Based and Checklist-Based Reading. In Proceedings of the Software Metrics, 10th international Symposium (September 11 - 17, 2004). METRICS. IEEE Computer Society, Washington, DC, 246-256. DOI= http://dx.doi.org/10.1109/METRICS.2004.3

[50] Cheng, B. and Jeffery, R. 1996. Comparing Inspection Strategies for Software Requirement Specifications. In Proceedings of the 1996 Australian Software Engineering Conference (July 14 - 18, 1996). ASWEC. IEEE Computer Society, Washington, DC, 203.

[51] Bernardez, B., Genero, M., Duran, A., and Toro, M. 2004. A Controlled Experiment for Evaluating a Metric-Based Reading Technique for Requirements Inspection. In Proceedings of the Software Metrics, 10th international Symposium (September

11 - 17, 2004). METRICS. IEEE Computer Society, Washington, DC, 257-268. DOI= http://dx.doi.org/10.1109/METRICS.2004.1

[52] Lanubile, F., Shull, F., and Basili, V. R. 1998. Experimenting with Error Abstraction in Requirements Documents. InProceedings of the 5th international Symposium on Software Metrics (March 20 - 21, 1998). METRICS. IEEE Computer Society, Washington, DC, 114.

[53] Haruhiko Kaiya, Hisayuki Horai, Motoshi Saeki: AGORA: Attributed Goal-Oriented Requirements Analysis Method. RE 2002: 13-22

[54] Hancock, B 1998. An Introduction to Qualitative Research. UK: Trentfocus Org.

[55] Kahn, R.L., Cannell, C.F. (1957), The Dynamics of Interviewing: Theory, Technique, and Cases, Wiley, New York, NY,.

[56] Siw Elisabeth Hove; Bente Anda, "Experiences from Conducting Semi-Structured Interviews in Empirical Software Engineering Research", 11th International Software Metrics Symposium METRICS 2005

[57] Carmel, E., Whitaker, R. D., and George, J. F. 1993. PD and joint application design: a transatlantic comparison. *Commun. ACM* 36, 6 (Jun. 1993), 40-48. DOI= http://doi.acm.org/10.1145/153571.163265

[58] A. Spangler, Cleanroom software engineering-plan your work and work your plan in small increments, IEEE Potentials, 15(4), 1996, p29 –32

[59] Langari, Z. and Pidduck, A. B. 2005. Quality, cleanroom and formal methods. *SIGSOFT Softw. Eng. Notes* 30, 4 (Jul. 2005), 1-5. DOI= http://doi.acm.org/10.1145/1082983.1083302

[60] D. Fetzer and J. Poore, "Using Box Structures with the Z Notation," Proceedings of the 25th Annual Hawaii International Conference on System Sciences, Vol. II--Software Technology Track, IEEE Computer Society Press, Los Alamitos, CA (January 1992).

[61] Menon, U., O'Grady, P.J., Gu, J.Z., Young, R.E. (1994), "Quality function deployment: an overview", in Syan, C.S., Menon, U. (Eds),Concurrent Engineering: Concepts, Implementation and Practice, Chapman & Hall, London, pp.91-9.

[62] Verma, D., R. Chilakapati, and B. Blanchard, "Quality Function Deployment (QFD): Integration of Logistics Requirements into the Mainstream System Design and Development Process," Proceedings, Annual Symposium, Society of Logistics Engineers, San Antonio, TX, August 1995

[63] Wood, C. 1998. Meeting Customer Needs Using Participatory Techniques. In *Proceedings of the Australasian Conference on Computer Human interaction* (November 29 - December 12, 1998). OZCHI. IEEE Computer Society, Washington, DC, 336.

[64] Kensing, F. and Blomberg, J. 1998. Participatory Design: Issues and Concerns. *Comput. Supported Coop. Work* 7, 3-4 (Jan. 1998), 167-185. [2] Participatory Design: issues and concerns

[65] Herlea, D.E. (1996), Users Involvement in the Requirements Engineering process in the Proceedings of KAW'96, Banff, Alberta, Canada

[66] Gregory, J. "Scandinavian Approaches to Participatory Design," International Journal of Engineering Education (19:1), Special Issue on Social Dimensions of Engineering Design, C. Dym, and L. Winner (eds.), 2003, pp. 62--74.

[67] Yau,S.S., Yeom, K., Gao, B., Li, L., Bae, D. An Object-Oriented Software Development Framework for Autonomous Decentralized Systems.Autonomous Decentralized Systems, 1995. Proceedings. ISADS 95., Second International Symposium on (1995), 405-411

[68] Kelly, D. and Shepard, T. 2001. A case study in the use of defect classification in inspections. In Proceedings of the 2001 Conference of the Centre For Advanced Studies on Collaborative Research (Toronto, Ontario, Canada, November 05 - 07, 2001). D. A. Stewart and J. H. Johnson, Eds. IBM Centre for Advanced Studies Conference. IBM Press, 7.

[69] Evans, I. 2004. A Practitioner's Guide to Software Test Design. By Lee Copeland. Published by Artech House, Norwood, MA, U.S.A., 2004. ISBN: 1-58053-791-X, 320 pages.: Book Reviews. Softw. Test. Verif. Reliab. 14, 4 (Dec. 2004), 283-284. DOI= http://dx.doi.org/10.1002/stvr.v14:4

[70] G. Vijayaraghavan and C. Kaner, 2003. Bug Taxonomies: Use Them to Generate Better Tests. In the Software Testing Analysis & Review Conference (STAR) East (Orlando, Florida, USA, 2003).

[71] Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., and Wong, M. 1992. Orthogonal Defect Classification-A Concept for In-Process Measurements. IEEE Trans. Softw. Eng. 18, 11 (Nov. 1992), 943-956. DOI= http://dx.doi.org/10.1109/32.177364

[72] Wagner, S. 2008. Defect classification and defect types revisited. In Proceedings of the 2008 Workshop on Defects in Large Software Systems (Seattle, Washington, July 20 - 20, 2008). DEFECTS '08. ACM, New York, NY, 39-40. DOI= http://doi.acm.org/10.1145/1390817.1390829

[73] Jane Huffman Hayes, Ashlee Holbrook, Inies Chemannoor, Dave Pruett, "Fault-Based Analysis: How History Can Help Improve Performance and Dependability Requirements for High Assurance Systems," accepted to Fifth International Workshop on Requirements for High Assurance Systems (RHAS), to be presented in Chicago, IL on November 8, 2005.

[74] Kelly, D. and Shepard, T. 2001. A case study in the use of defect classification in inspections. In Proceedings of the 2001 Conference of the Centre For Advanced Studies on Collaborative Research (Toronto, Ontario, Canada, November 05 - 07, 2001). D. A. Stewart and J. H. Johnson, Eds. IBM Centre for Advanced Studies Conference. IBM Press, 7.

[75] IEEE (1993). IEEE standard classification for software anomalies. IEEE Standard 1044-1993.

[76] Grady, R. B. 1992 Practical Software Metrics for Project Management and Process Improvement. Prentice-Hall, Inc.

[77] Christel, M. & Kang, K. 1992. Issues in Requirements Elicitation. (CMU/SEI-92-TR-012, ADA258932). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University

[78] Kaner, C., 1998. The Performance of the N-Fold Requirement Inspection Method., Requirements Engineering Journal, V. 2, N. 2, Month , pp. 114-116

[79] 2004 A Guide to the Project Management Body of Knowledge (PMBOK Guides). Project Management Institute.

[80] Trochim, W. and Donnelly, J.P. (2007). The Research Methods Knowledge Base. 3rd edition. Thomson Publishing, Mason, OH.

[81] Guba, E & Lincoln, Y 1981, Effective evaluation: Improving the usefulness of evaluation results through responsive and naturalistic approaches, Jossey-Bass, San Francisco.

[82] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. 2000 Experimentation in Software Engineering: an Introduction. Kluwer Academic Publishers.

[83] Seidel, J. (1991). Qualitative Data Analysis, Qualis Research, DOI = http//www.qualisresearch.com

Appendix A: Interview Questionnaires

- What are requirements level defect types and reasons for defects related to Elicitation, Analysis and Negotiation (E and A&N) phases that can cause major rework in later stages of SDLC?
    - What are the most common defects types reported by research based on E and A&N?
    - ***What are the most common defects types and reasons for defects originating from E and A&N as reported from Swedish and Pakistani software companies respectively?***

1. What types of defects you find mostly that originate from requirements E and A&N phase in your projects.
    - What types of defect are identified by requirements analysts during requirements analysis process? For example it could be inconsistencies in requirements, unclearness, missing, and incomplete requirements.
    - What types of requirements defects are reported by testers?
    - When you have release your project then what kinds of defects customer reports?
2. what do you think about the origination of these defects (E or A&N phase)
3. Who reports major defects? (customer, analyst, requirements validation team, tester or system maintainer)
4. Do these defects were found first time?
5. Do these defects are repeatedly found?
6. What are reasons for these defects? (like incomplete requirements, unclear requirements) there may be some risks that can be a reason for defects or may be a defect type itself. Questions regarding them is given below
    - Do all requirements engineers are very interested in spending time for the RE process or they think that they have understood all user requirements?
    - Do all users are agree to give detailed requirements?
    - Do the developer Gold plating? If yes then is it expectable for customer or he always refuse?
    - Do the users request changes frequently or seldom (creeping user requirements)?
    - If users request changes then does the developers response to that change?
    - Do the elicitators gather detailed requirements or just get main idea of the product and leave the detail requirements gathering task upon developers during implementation phase?
    - When requirements specification document is frozen?
    - Do requirements are changed frequently?
    - Does your customer always want you to make changes?

7. How do you categorize these defects? Is there any defect taxonomy?
8. Are there some defect types beyond this defect taxonomy?
9. In which stage of SDLC you keep focus in finding defects and fix them? (RE, implementation, Testing or maintenance)
10. Do you believe in finding and fixing the defects during testing phase? If yes then why you don't believe in DP in early stages of SDLC?
11. What is percentage of defects found that have been identified by customers?
12. What is percentage of defects found that have been identified during requirements

validation?

- *Find possible rework caused by each defect?*

13. How do you differentiate between major and minor defects found in your projects?
14. If a major defect (like missing requirements) is identified during requirements engineering process then can you estimate the possible rework cause by that defect if it is identified and fixed during testing or maintenance phases of SDLC (Low, Medium or High)?

- ***Are there any major differences between Swedish and Pakistani companies in terms of types and rate of defects originating from E and A&N phases?***

1. What types of defects you identify during V and V process those originate from E and A&N phases?
2. In SDLC, at what stage (RE, Design, implementation or testing) where you mostly identify defects and fix them?
3. Does you organization believe in DP strategy or just believe in find and fix the defects.
4. How many people are assigned for elicitation process?
5. How many people are assigned for analysis process?
6. How many people are assigned for validation process?
7. What is skill level of your employees? (Bad, Satisfactory, Very good or Excellent)
8. Do your employees have special training for the RE process?
9. How much experience your employees have in the RE process (1, 2, 3, 4, 5 years or more)?
10. What is rate of defects identification (during verification and validation activities) that originates from E and A&N phases?

- What DPT is associated with each defect type that is being practicing in industry based on E and A&N phases?
  - ***What is appropriate DPT for each defect type?***
  - What are the problems in existing DPT (s)?

11. When you find a defect originating from E or A&N phase then what kinds of measures you adopt?
12. Are there any preventive actions (like change in technique, process or use new tool) that you take against them to avoid them to reoccur? If yes then please tell us about it.
13. Are you following any DP approach during E or A&N phases (like clean room, JAD)?
14. What techniques for defect identification you have been using in requirements analysis, specification and validation processes?

Note: *Questions in boxes are interview questions related to research questions written above each research question.*

Appendix B: Boris Bug Taxonomy (BBT)

(Adopted from [32])

1xxx: FUNCTIONAL BUGS: REQUIREMENTS AND FEATURES: bugs having to do with requirements as specified or as implemented.
11xx: REQUIREMENTS INCORRECT: the requirement or a part of it is incorrect.
111x: Incorrect: requirement is wrong.
112x: Undesirable: requirement is correct as stated but it is not desirable.
113x: Not needed: requirement is not needed.
12xx: LOGIC: the requirement is illogical or unreasonable.
121x: Illogical: illogical, usually because of a self–contradiction which can be exposed by a logical analysis of cases.
122x: Unreasonable: logical and consistent but unreasonable with respect to the environment and/or budgetary and time constraints.
123x: Unachievable: requirement fundamentally impossible or cannot be achieved under existing constraints.
124x: Inconsistent, incompatible: requirement is inconsistent with other requirements or with the environment.
1242: Internal: the inconsistency is evident within the specified component.
1244: External: the inconsistency is with external (to the component) components or the environment.
1248: Configuration sensitivity: the incompatibility is with one or more configurations (hardware, software, operating system) in which the component is expected to work.
13xx: COMPLETENESS: the requirement as specified is either ambiguous, incomplete. or overly specified.
131x: Incomplete: the specification is incomplete; cases, features, variations or attributes are not specified and therefore not implemented.
132x: Missing, unspecified: the entire requirement is missing.
133x: Duplicated, overlapped: specified requirement totally or partially overlaps another requirement either already implemented or specified elsewhere.
134x: Overly generalized: requirement as specified is correct and consistent but is overly generalized (e.g., too powerful) for the application.
137x: Not downward compatible: requirement as specified will mean that objects created or manipulated by prior versions can either not be processed by this version or will be incorrectly processed.
138x: Insufficiently extendable: requirement as specified cannot be expanded in ways that are likely to be needed—important hooks are left out of specification.
14xx: VERIFIABILITY: specification bugs having to do with verifying that the requirement was correctly or incorrectly implemented.
141x: Unverifiable: the requirement, if implemented, cannot be verified by any means or within available time and budget. For example, it is possible to design a test, but the outcome of the test cannot be verified as correct or incorrect.
142x: Untestable: it is not possible to design and/or execute tests that will verify the requirement. Untestable is stronger than unverifiable.
15xx: PRESENTATION: bugs in the presentation or documentation of requirements. The requirements are presumed to be correct, but the form in which they are presented is not. This can be important for test design automation systems, which demand specific formats.
152x: Presentation, documentation: general presentation, documentation, format, media, etc.
153x: Standards: presentation violates standards for requirements.
16xx: REQUIREMENT CHANGES: requirements, whether or not correct, have been changed between the time programming started and testing ended.
162x: Features: requirement changes concerned with features.

1621: Feature added: a new feature has been added.
1622: Feature deleted: previously required feature deleted.
1623: Feature changed: significant changes to feature, other than changes in cases.
163x: Cases: cases within a feature have been changed. Feature itself is not significantly modified except for cases.
1631: Cases added.
1632: Cases deleted.
1633: Cases changed: processing or treatment of specific case(s) changed.
164x: Domain changes: input data domain modified: e.g., boundary changes, closure, treatment.
165x: User messages and diagnostics: changes in text, content, or conditions under which user prompts, warning, error messages, etc. are produced.
166x: Internal interfaces: direct internal interfaces such as call sequences, or indirect interfaces (e.g., via data structures) have been changed.
167x: External interfaces: external interfaces, such as device drivers, protocols, etc. have been changed.
168x: Performance and timing: changes to performance requirements (e.g., throughput) and/or timings.

Appendix C: Improvements Validation (Company C)

S.No.	Questions
1	***What do you think about start-up meeting and the things that should be discussed in the meeting? Would it be proved effective in preventing requirements level defects?*** Yes, the startup meeting will help to improve the RE process and minimize RE level defects and the content of this meeting are beneficial prior to the RE process initialization. Its better if you plan these content in advance but it will take time
2	***Should it be (start-up meeting) beneficial for improving overall RE process and minimizing requirements defects?*** Yes, by considering things that you decided to discuss in start-up meeting is related to improvement in the RE process and helps in minimizing requirements defects.
5	***Do you think the use of maximum number of scenarios for requirements help to overcome missing and incompleteness in requirements?*** Scenarios did not cover all type of defects but to identify or detect incompleteness and missing requirements scenarios can be useful
6	***Do you think that use of IEEE standards for specifying requirements helps to improve SRS quality?*** Yes, use of standards will also help to overcome defects related to documentation of requirements.
8	***Do you think that the information obtained from classification of defects proves to be valuable in start-up meeting?*** Yes, properly classified defects help to understand the types of defects which is beneficial for improvements.
9	***Do you think training, experience and domain knowledge can improve elicitation process?*** Yes, it is necessary that people involved in requirements elicitaion should be experienced and have completed domain knowledge to understand their customer's requirements. They must be some training to make this process efficient.
10	***Do you think an SRS that is unambiguous, complete, verifiable, consistent, modifiable, and traceable is a quality SRS?*** Yes, these qualities must exist for a good SRS and I think using standards like IEEE for specification will help to achieve these SRS attributes.
11	***Do you think Four-fold inspection is an adoptable requirements analysis technique as compared to N-fold inspection?*** Yes, it is efficient but it depends upon the size of SRS and the type of project. For medium size project up to 200 requirements it would be beneficial but if we talk about product with 10, 000 requirements then four-fold could be limited.
12	***Do you think that defect detection rate of Four-fold inspection would be low, medium or high?*** In case of single team the defect detection rate is 27% but in case of four fold inspection this detection rate would be 42% to 43%.
13	***Do you think Four-fold inspection would catch requirements defects nearly equal to N-fold inspection (where N=4)?*** I think four-fold technique would reports defects more than N fold for N=4 because using divide and conqueror rule
14	***Do you think Four-fold inspection is a cost effective inspection technique as compared to N-fold inspection?***

	Although resources are equally consumed but less cost and time is used as compared to N-Fold (N=4)
15	***Do you think the combination of our proposed requirements defect taxonomy and scenario based reading (defect based reading) technique proposed in Four-fold inspection can identify good number of major requirements defects?*** Yes, it is better to use taxonomy and scenarios in combination with four-fold. The reason for this is when you inspect a document in scenario based reading of requirements which mean you can also see them as use cases from different perspectives with respect to its usage. Second you also visualize requirements from real usage of end users. This will also reduce their ripple affect into later stages if you tackle requirements defects at RE level.
16	***What do you think about our proposed solution (list of recommendations, Four-fold inspection technique, and requirements defect taxonomy) would prove a good attempt to prevent requirements level defects and to minimize rework caused by those defects in later stages of SDLC?*** Yes, it is helpful if you consider things in advance to cause lack in requirements and to reduce their ripple effect in later stages of SDLC. It is better to initially clear and fix things at requirements level.
17	***We have observed during industrial research that most of the interviewees don't know about defect prevention techniques (DPTs) and methods. We have given detail of them in our thesis. What do you think if the companies would get familiar with such DPTs then can they prevent more requirements level defects?*** ??
18	***During our research we found that missing quality attributes unwanted/unnecessary requirements, misunderstood requirements, omissions; missing functional requirements are the most common and major defect types. It means that more effort and time should be spent on these defect types to minimize avoidable rework that is done when preceding defect types are identified and fixed down the development road. What do you think our above suggestion is would be good step in preventing the requirements level defects? Or you have any suggestion?*** ??
19	***Do you have any suggestions to improve list of recommendation or four fold inspections?*** ??

Appendix D: Improvements Validation (Company D)

S.No.	Questions
1	***What do you think about start-up meeting and the things that should be discussed in the meeting? Would it be proved effective in preventing requirements level defects?*** Recently we had a lecture in our company where the lecturer discussed the similar problems like risks associated to SDLC. He also stressed to do care of risks. So definitely it is important to care of requirements level risks and obliviously by considering these risks the defects associated with requirements will be minimized.
2	***Should it be (start-up meeting) beneficial for improving overall RE process and minimizing requirements defects?*** A part from the contents that should be discussed in start-up meeting. Users also have some tacit assumption about requirements which he/she think that requirements analyst should consider it by default. So it should also be discussed in this meeting.
5	***Do you think the use of maximum number of scenarios for requirements help to overcome missing and incompleteness in requirements?*** Yes, it is better to uncover missing and incomplete requirements through the use of scenarios.
6	***Do you think that use of IEEE standards for specifying requirements helps to improve SRS quality?*** Yes, use of standards for specifying requirements helps to improve the SRS quality.
8	***Do you think that the information obtained from classification of defects proves to be valuable in start-up meeting?*** Yes, properly classified defects help to understand the types of defects which is beneficial for improvements.
9	***Do you think training, experience and domain knowledge can improve elicitation process?*** Yes, these things are necessary for an efficient requirements elicitation process.
10	***Do you think an SRS that is unambiguous, complete, verifiable, consistent, modifiable, and traceable is a quality SRS?*** Yes, these qualities must exist for a good SRS and I think using standards like IEEE for specification will help to achieve these SRS attributes.
11	***Do you think Four-fold inspection is an adoptable requirements analysis technique as compared to N-fold inspection?*** I think it is an adoptable requirements analysis technique which can work efficiently as compare to N-fold inspection having N = 4 and it can save time as well.
12	***Do you think that defect detection rate of Four-fold inspection would be low, medium or high?*** I think four-fold inspection will identify defects equal to N-fold inspection if you will put N=4.
13	***Do you think Four-fold inspection would catch requirements defects nearly equal to N-fold inspection (where N=4)?*** Yes, it will work but inspector must be trained or expert in SBR technique and GORA technique. The use of defect taxonomy that you have proposed is also a good choice.
14	***Do you think Four-fold inspection is a cost effective inspection technique as compared to N-fold inspection?***

	Yes, it is cost effective because teams are only four to identify maximum number of defects but training and expertise are required to get better results.
15	***Do you think the combination of our proposed requirements defect taxonomy and scenario based reading (defect based reading) technique proposed in Four-fold inspection can identify good number of major requirements defects?*** Yes, I think it is a good attempt to minimize requirements levels defects. The use of scenario based reading and GORA technique seems a good combination with four fold inspection. I will again repeat that training is very important to get good results.
16	***What do you think about our proposed solution (i.e. list of recommendations, Four-fold inspection technique, and requirements defect taxonomy) would prove a good attempt to prevent requirements level defects and to minimize rework caused by those defects in later stages of SDLC?*** Definitely your proposed solutions are a good attempt to solve the problems that you have mentioned in your thesis.
17	***We have observed during industrial research that most of the interviewees don't know about defect prevention techniques (DPTs) and methods. We have given detail of them in our thesis. What do you think if the companies would get familiar with such DPTs then can they prevent more requirements level defects?*** Yes you are right that people in companies don't know about the DPTs but if you see at company level the perhaps they know them. I think if software developer or analyst get familiar with DPTs top prevent requirements level defects then definitely they would get good results. It is good attempt by you that you have given description of all DPTs or DP methods in your thesis.
18	***During our research we found that missing quality attributes unwanted/unnecessary requirements, misunderstood requirements, omissions; missing functional requirements are the most common and major defect types. It means that more effort and time should be spent on these defect types to minimize avoidable rework that is done when preceding defect types are identified and fixed down the development road. What do you think our above suggestion is would be good step in preventing the requirements level defects? Or you have any suggestion?*** Yes, definitely it will help in preventing requirements level defects if you will put more effort on above mentioned defects.
19	***Do you have any suggestions to improve list of recommendation or four fold inspections?*** I have no more suggestions except one that is training in four fold inspection, SBR, and GORA technique.

Appendix E: Common Requirements Defect Types and their Reasons

Ambiguity - ambiguity in requirements in early stages leads to confusion, misinterpretation and misunderstanding of requirements. Different meanings of requirement construct different interpretation about it. This happens only if the requirements or part of it is too ambiguous to understand or requirements are not initially clear. To deal with the problem there must be some common understanding of each and every requirement among the users of those requirements. Secondly, issues related to requirements must be resolved at early stages to avoid rework. Ambiguity happens because of

- Improper translation of requirements,
- Requirements are not initially clear
- Requirements are poorly described or defined
- Incomplete requirements description
- Use of third party software also leads to misunderstanding, confusion and associated defects [31] [33].

Inconsistencies – inconsistency in requirements is the most common issues exist because of conflicts or contradiction among requirements. During requirements analysis, conflicts must be resolved. Inconsistencies in requirements arise as a result of varying customer's demands, different viewpoint of customers about requirements, disagreements among customers and mistakes or omission in requirements [7]. Requirements are crosschecked to identifying problems and conflicts are resolved by organizing a conflicts resolution meeting with the customers [7].

Omissions – omission occurs as a result of errors or deficiencies in requirements which make the requirements incomplete. During elicitation or specification process, it is observed that most of the requirement analysts use their experience to elicit customer requirements. If requirements are not immediately written down as they are elicited then there might be a chance that some requirements or important information could be left out. This lack of information or unwritten requirements leads to omission in requirements. Omission occur as a result of

- Poor elicitation process
- Lack of experience of analyst
- Requirements that were elicited but not written down
- Tacit requirements

Unwanted/unnecessary requirements - these defects are not mostly generated by customer but the development organization uses the concept of gold plating to gain customer compassion. Adding extra features or functionality to products although not demanded by customer may lead to many problems. Unwanted or unnecessary requirements occur because of

- Gold plating
- Extraneous requirements
- Wrong assumptions about requirements

Volatility – good software always welcome new changes in requirements but if the requirement change over time in development phase then there could be some problem with the requirements. Volatility in requirements occurs because of following some factors [7].

- Customers little or minute knowledge about system can make the requirements volatile
- Chance in system environment

- Technology change
- Changing customers priorities
- Change in organizational process or structure

Clarity – in requirements also leads to undesirable requirements because customers themselves are not clear about the system. This problem occurs because of incorrect requirements description, wrong requirements assumptions, requirements badly expressed or omitted requirements during requirements elicitation.

Incorrect fact – deals with feasibility of requirements where a requirement is correct but it is not feasible or suitable to implement it under certain environment and under particular condition. This usually comes under infeasible requirements but the problem in requirement is the wrong behavior it under certain condition of system. To deal with this problem, stakeholder must be consulted to make modification about the requirements and make it realistic. [7]

Missing requirements – is the most commonly occurring problems consisting of missing features, functionality, missing requirement or part of it, missing quality attributes, and missing environment etc. Missing requirements occurs because of

- Poor elicitation process
- Poor domain knowledge
- Wrong requirements specification
- Incomplete requirements.

Unstable Requirements – directly deals with the volatility of requirements. Too many changes in requirements make the requirement unstable. Unstable requirements create the problem of decision making regarding freezing SRS and to make the requirements part of a certain release due to volatility.