Alexandria Engineering Journal (2014) 53, 513–527

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

Detecting defects in software requirements specification

Amira A. Alshazly [a,*], Ahmed M. Elfatatry [a], Mohamed S. Abougabal [b]

[a] Information Technology Dept., Institute of Graduate Studies & Research (IGSR), Alexandria University, Alexandria, Egypt
[b] Computer and Systems Engineering Dept., Faculty of Eng., Alexandria University, Alexandria, Egypt

Received 12 February 2014; revised 18 May 2014; accepted 1 June 2014
Available online 28 June 2014

KEYWORDS

Defects errors;
Taxonomy classification;
Inspection detection;
Methods for SQA and V&V;
Software requirements specification quality

Abstract This research is concerned with detecting defects in software requirements specification. Motivated by both the problem of producing reliable requirements and the limitations of existing taxonomies to provide a satisfactory level of information about defects in the requirements phase, we focus on providing a better tool for requirements analysts. Only few attempts have been made to classify defects and defect detection techniques. Scattered knowledge about defects and defect detection techniques needs compilation and re-evaluation in order to enhance the ability to discover defects in the requirements phase. Toward this end, this work presents a taxonomy of requirements defects and the causes of their occurrences. The purpose is to reach a comprehensive understanding of both the sources of the problem and the solutions of possible defects and defect detection techniques. The taxonomy's design is based on the analysis of each defect and its sources. In addition, this paper proposes a combined-reading technique for defects in requirements. The proposed technique avoids the shortcomings of other reading techniques. The result of applying the recommendations of this work specifically improves the quality of the requirements specification and generally software quality.

1. Introduction

The requirements phase is the most critical phase of the software development life cycle (SDLC). Wrong or missing requirements lead to wrong or incomplete product; no matter how good the subsequent phases are. The quality of the requirements phase affects the overall quality of the subsequent phases and hence, the software product. Writing good software requirements specification (SRS) is an important determinant of software quality [1]. The SRS document defines the capabilities of the provided software [2]. Therefore, if an analyst or a developer does not share the same understanding about requirements, the outcome of the development process will not satisfy the customers' needs [3]. The more progress in the software development life cycle (SDLC), the more defects will emerge. Consequently, the earlier the detection of requirements defects, the much money and time of rework can be saved. To achieve this, it is important to study defects

* Corresponding author. Tel.: +20 1062642986.
E-mail addresses: amira.alshazly@gmail.com, amira_alshazly@yahoo.com (A.A. Alshazly).

Peer review under responsibility of Faculty of Engineering, Alexandria University.

Production and hosting by Elsevier

1110-0168 © 2014 Production and hosting by Elsevier B.V. on behalf of Faculty of Engineering, Alexandria University.
http://dx.doi.org/10.1016/j.aej.2014.06.001

in the requirements phase and defect detection techniques, especially in the SRS document.

There is a need to organize the scattered information about defects and defect detection techniques in order to put such information in a framework that shows areas of shortages. The organized knowledge is presented in the form of a taxonomy. While a number of techniques have been proposed for detecting defects in the requirements phase, little work has been done in analyzing and evaluating the suitability of available techniques for different situations in the requirements phase [4]. Previous taxonomies of defects did not correlate the defects in the requirements phase and the reasons for their occurrence, as discussed in Section 3.

This work is motivated by the limitations of existing taxonomies to achieve a satisfactory level of information about defects in the requirements phase. These limitations were overcome by proposing taxonomy of defects in requirements specification. The taxonomy correlates each defect and the reasons for its occurrence in SRS documents. Also, the paper compares the effectiveness of defect detection techniques (reading techniques) in the requirements phase to determine the efficiency and effectiveness of each reading technique. In addition, this paper provides a combined-reading technique that is an enhanced reading technique. The proposed technique avoids the shortcomings of other reading techniques. It helps requirements engineers in detecting defects efficiently and correcting them in the requirements phase. Therefore, the quality of software could be enhanced.

First, a literature survey is carried out to reveal the current state-of-the-art of research in requirements defects and existing classifications. Another survey has been conducted on the proposed requirements defect detection techniques (reading techniques). Then, case-studies were applied on a number of SRS documents to produce the taxonomy of defect types and to determine the efficiency and effectiveness of the reading techniques. The results of the taxonomy and the comparison of the reading techniques were analyzed to present a combined-reading technique. The combined reading technique has been evaluated using four case-studies.

The paper is organized as follows. A literature review of defects classification that can occur in requirements and a survey of the requirements reading techniques are provided in section two. Classification of defects is studied in section three. The defect detection techniques are compared in section four. A taxonomy of requirements defects is proposed in section five. The combined-reading technique is presented in section six. Both the taxonomy of requirements defects and the combined-reading technique are evaluated in section seven. Finally, the work is concluded in Section 8.

2. Related work

The previous work concerning defect classification is presented in this section and then a survey of the requirements' defect-detection techniques (reading techniques) is discussed.

2.1. Defects classifications

In this section, three taxonomies related to requirements defects are discussed. Two taxonomies are related to defects, which are the IEEE Std. 1028-1997 that classified the anomaly classes [5], and the classification of defect types in requirement specifications presented in [6]. The third is concerned with requirements' errors that cause defects [7].

2.1.1. The IEEE Std. 1028-1997 categorization of anomaly classes

The IEEE Std. 1028-1997 categorized anomaly classes as: missing, extra (superfluous), ambiguous, inconsistent, improvement desirable, not conforming to standards, risk-prone, factually incorrect, not-implementable (for example because of system constraints or time constraints), and editorial [5].

2.1.2. Classification of defect types in requirements specifications

Margarido et al. [6] classified defect types in requirements specifications by studying the literature review of defect taxonomies, deducing the adequacy of the classifiers to classify requirements defects, and identifying the frequency of the defects classifiers. They classified requirements defects into: missing or incomplete, incorrect information, inconsistent, ambiguous or unclear, misplaced, infeasible or non-verifiable, redundant or duplicate, typo or formatting, and not relevant or extraneous.

2.1.3. Classification of requirements' errors

Walia et al. [7] developed a taxonomy of requirements' errors in the requirements phase. They identified and collected

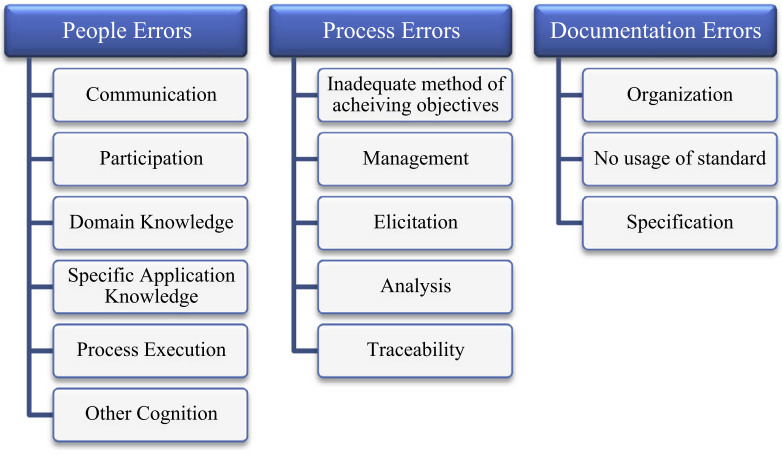

Figure 1 Classification of requirements' errors [7].

fourteen types of errors (sources of defects) from literature survey of software engineering, psychology and human cognitive fields. Then, they categorized errors into three high-level classes of requirements' errors: people errors, process errors, and documentation errors, as follows (see Fig. 1).

- People errors include errors resulting from people involved in requirements preparation.
- Process errors include errors that occur due to inadequate requirement engineering process, and selecting wrong means of achieving goals and objectives.
- Documentation errors include errors that occur due to incorrect organization and specification of requirements, regardless of whether the requirements author understood the requirements correctly or not.

The key advantage of the requirement error taxonomy is to cover errors that appear in requirements engineering steps. This includes requirements elicitation, requirements analysis and negotiation, requirements specification and requirements management which help in the requirements' verification. Whereas the main drawback of this classification is that it does not relate the types of defects to the causes leading to their appearance.

2.2. Requirements defect-detection techniques

Concerning defect detection, several comparisons were made between the requirements' defect detection techniques, such as the comparison between Perspective-Based-Reading techniques (PBR) with Checklist-Based-Reading technique (CBR) as [8,9]. A number of researches have compared the Perspective-Based-Reading techniques (PBR) with Checklist-Based-Reading technique (CBR) and ad-hoc based reading technique such as the work described in [10–12]. Others compared the Scenario-Based-Reading techniques (SBR) with Checklist-Based-Reading technique (CBR) and ad-hoc based reading technique such as in [13,14]. Also, [15] presented a comparison between the Usage-Based-Reading techniques (UBR) and Checklist-Based-Reading technique (CBR).

In industry, several studies and surveys were conducted on the requirements validation techniques (RVTs). Sulehri [16] conducted interviews and a survey of requirements engineering departments' employees to judge the best performance of the requirements validation techniques (RVTs) in six software companies in Sweden. Concerning requirements inspection techniques, the six companies used one of two reading techniques which are Ad-hoc based reading and Checklist-Based-Reading (CBR). It was found that the requirements inspection techniques were generally very effective but they were only useful for large-scale software organizations that have a large number of human resources.

Saqi and Ahmed [17] carried out a study on six companies from two countries regarding requirements validation. These companies used one of two reading techniques which are ad-hoc based reading and Checklist-Based-Reading (CBR). They reached the same results of Sulehri's study.

Concluding the industrial practices, in spite of conducting these studies on the requirements validation techniques (RVTs), no comparison has been made on the effectiveness of the inspection techniques. The objective of the following sub-sections is to review the defect detection techniques (reading techniques) in requirements specification documents. The purpose is to identify the goal, characteristics, advantages and disadvantages, procedures of using each technique, and the responsibility of the inspector in each technique.

2.2.1. Checklist-Based-Reading technique (CBR)

In Checklist-Based-Reading technique, the reviewer gets a checklist that is expressed in the form of questions or statements in order to search for a special kind of faults [18]. Inspectors read the document during answering a list of yes/no questions that are based on their previous knowledge of typical defects [10]. The checklist should not exceed a page for each document [19]. The CBR technique is a non-systematic technique [8] because it does not provide instructions on how the inspection can be performed. In CBR technique, the reader is responsible for all the inspection processes and finding all possible defects.

The Checklist-Based-Reading technique has the advantage of providing guidance on what is inspected for [11], yet the disadvantages are listed below.

- The questions are often general. They are usually collected from previous projects, literature, or other organizations, and not sufficiently tailored to a particular development environment [20].
- It does not tell the inspector how to ensure the software quality attribute [20].
- It does not provide instructions on how the inspection can be performed or how to use a checklist [20].
- Checklist questions are usually limited to specific types of defects. Therefore, inspectors may not focus on defect types that have not been previously detected. Consequently, they may entirely miss classes of defects [20].
- All inspectors have the same checklist and should answer all the questions [21] which lead to repeating and wasting inspector's effort, as a result of overlapping work of inspectors.

2.2.2. Defect-Based-Reading technique (DBR)

Defect-Based-Reading technique concentrates on specific defect types in requirements, where requirements are expressed by state machine notation called Software Cost Reduction (SCR) [22]. The DBR technique is carried out by the following steps [19].

1. Defects are classified.
2. A set of questions is posed for each defect class.
3. Scenarios which are derived from checklists are built.
4. Each scenario is assigned to a reviewer to detect a particular type of defects.

This technique has the advantage of providing guidance for inspector about "What and How" it is inspected for [11]. While its drawback is that the scenarios limit the attention of the inspector to the detection of particular defects that are defined by the custom guidance [20]. As a result, other types of defects will not be detected unlike the checklist technique which search for any defects in the documents [21].

2.2.3. Perspective-Based-Reading technique (PBR)

Perspective-Based-Reading techniques concentrate on examining artifacts from different perspectives of the users of software documents in order to improve efficiency by minimizing the overlap among the faults found by the reviewers [15]. The PBR techniques are carried out by identifying perspectives that depend upon the roles people have within the software development or maintenance process. For each perspective, either one or multiple scenarios are defined that consist of repeatable activities performed by the inspector, and questions answered by the inspector. This is done in order to increase the understanding of the software product from a particular perspective. It helps the inspector to identify defects [20].

The PBR techniques have the following advantages [8].

- PBR is more effective, systematic, focused, goal-oriented, customizable and transferable via training because reviewers inspect software documents from specific stakeholders' perspectives concerning SRS to verify its correctness.
- Less overlap in defect detection than CBR technique.
- PBR is better than CBR for reviewers who are less familiar with software domain.
- To conclude this section, all Scenario-Based-Reading (SBR) techniques (DBR and PBR techniques) share the following factors.
- They are systematic techniques [8] because they provide a procedure for how to read a document.
- The reader is only responsible for detecting defects that can be found from a particular point of view, such as analyst or designer perspectives, or special defect type as omission or incorrectness.

3. Comparative analysis of defect classifications

The following results were found by comparing the classifications of defects in the requirements phase presented in Section 2.1.

- The objective of the IEEE categorization of anomaly classes is to classify types of defects in SDLC to improve the quality of software. The objective of the classification of Margarido et al. is to classify defect types in SRS documents in order to build checklists to support requirements' reviewers, whereas the objective of Walia's classification is to classify errors that could cause defects in requirements phase generally to support the prevention and detection of errors.
- Both classifications of Margarido et al. and Walia et al. used the empirical research approaches to design their classification while the methodology that has been used to conduct the IEEE categorization of anomaly classes is not specified.
- The IEEE categorization does not focus on any phase or any type of document. It works on all SDLC phases. While the classification of Margarido et al. is concerned with the requirements phase and specifically with the SRS documents. Walia's classification of requirements' errors did not focus on any type of requirement documents.
- The IEEE categorization of anomaly classes did not define the characteristics of anomalies. It does not specify when and why an anomaly occurs or how to discover its existence, while the classification of Margarido et al. of defect types defined each of them and provided an example on each of them, but they did not specify the causes of these defects. Also, Walia's classification of requirements' errors defined each requirement error and gave an example on each of them but they did not relate types of defects and errors leading to their appearance.
- Both classifications of Margarido et al. and Walia et al. used the root-cause-analysis method for problem solving, taking into account the difference in research objective and problem. With regard to the IEEE categorization of anomaly classes, the used problem solving method is not specified.

To conclude this section, the IEEE classification focused on the defects of all phases in the SDLC. The classification of Margarido et al. is based on listing defects, defining each of them and giving a number of examples but did not specify the causes of these defects. The requirements' errors taxonomy of Walia et al., covered errors that appear in most of requirements engineering steps but did not correlate the types of defects and errors causing them. Thus, there is a need to find the relationship between the types of defects, and the errors causing them. Specifying when and why defects occur, or how to note its existence via scientific approach. Knowing the nature of each defect will help detection and prevention of requirement defects.

4. Comparative analysis of reading techniques

This section compares the effectiveness of defect detection techniques (reading techniques), and the aim is to assist inspectors in choosing effective and efficient techniques in order to guarantee the SRS documents' quality.

4.1. Experiment

To achieve that, an experiment has been carried out to compare the effectiveness of defect detection techniques (CBR, PBR and DBR), and to determine the ability of each technique to detect defects in terms of the type of defect and its source. The hypothesis to be investigated in the experiment is as follows.

H: Defect detection techniques vary in terms of their ability to detect each defect based on the source causing its occurrence.

To test this hypothesis, each defect detection technique has been applied to three SRS documents listed below.

- The *Automated Teller Machine network* (*ATM*). A system that supports computerized banking network which work together with the bank computer to dispense money for ATM cash-card carrier [23].
- The *Online National Election Voting System* (*ONEV*) is a system to provide information about the candidates before the date of the election, and to conduct the voting process at election stations on the election date [24].

- The *web publishing system* is a system for the editor of a regional historical society. The system assists the editor in the automation of an article reviewing and publishing process [25].

The detected-defects are classified into the type of defect, the main-reason for their appearance and then classified further in the sub-reason for their appearance. Then the detected-defects by each technique were compared with the defects detected by other techniques, as shown in Table 1. The comparison has been performed on the following aspects: the total number of detected-defects, the distribution of defects on their source of occurrence, and both the common and different detected-defects between each pair of techniques. Thereby, the effectiveness of defect detection techniques (reading techniques) has been compared.

4.2. Results

(See Table 1).

4.3. Analysis of results

Comparing the Checklist-Based-Reading technique to the Scenario-Based-Reading techniques, the following has been observed.

- According to the total number of the detected-defects, the PBR is more effective than DBR and CBR, while DBR is more effective than CBR.
- According to the type of the detected-defects, the following were concluded.
  - In case of ambiguous and omission defects, the PBR is more effective than DBR and CBR while DBR is more effective than CBR.
  - In case of inconsistent defects, the PBR is more effective than DBR and CBR whereas the techniques CBR and DBR are equal in effectiveness.
  - Concerning the incorrect defects, the DBR is more effective than CBR and PBR while CBR is more effective than PBR.
  - With regard to the superfluous defects, the CBR and DBR are equal in their effectiveness. While the PBR could not find any superfluous defects.

5. Proposed taxonomy of requirements defects

In this section, a taxonomy of requirements' defects and causes is presented. An empirical approach has been used to prepare the proposed taxonomy. In order to build the taxonomy, it is necessary to identify the factors contributing to the formation of the SRS document, determine the types of defects that occur in SRS documents, and the errors that may cause such defects.

Factors contributing to the formation of an SRS document are: SRS authors, the process of requirements analysis, and the process of writing the SRS document in documentation form.

A review of the literature reveals that certain types of defects have to be excluded for the following reasons.

- The excluded defects from the IEEE classification of anomaly classes are the "improvement desirable anomaly" and the "editorial anomaly". The "improvement desirable anomaly" is a defect that may occur in a design or a code which needs to be revised. Such defect type cannot occur in the requirements document as the requirements document is only a description of customer needs. The requirements document can be written in an ambiguous, inconsistent way or in a way that is not confirming to standards. These defects can be rewritten or improved in the rework (removing requirements defects). Based on the above, it is unacceptable that the requirements document contains an "improvement-desirable" defect. Moreover, the "editorial anomaly" is a typo or formatting error that can cause defects such as ambiguous, inconsistent or incorrect requirements, yet it is not a defect in itself.
- The excluded defects from the classification of Margarido et al. of defect types in requirements' specifications are "misplaced", "redundant" and "typo or formatting". "Misplaced" cannot be considered a defect type. For instance, if a requirement is misplaced in any section of the requirements' specification document, it will be considered a requirement error that may cause defects such as ambiguous, inconsistent or incorrect information. Furthermore, "redundancy" cannot be accepted as a defect. It could be considered a requirement error because it can cause defects such as ambiguous and inconsistent. Also, "typo" or "formatting" cannot be regarded as a defect. "Typo or formatting" class is a requirement error because it can cause defects such as ambiguous, inconsistent or incorrect information.

Hence, we specify the defects that can occur in SRS documents as: omission, ambiguous, inconsistent, superfluous, incorrect, not-conforming to standards, not-implementable and risk-prone. To reach an acceptable definition for each of the defect type, defect definitions in other SDLC phases have been reviewed in [26,27], p. 182] and [28] and the analysis of literature review for the types of defects in the requirements phase are reviewed. The description of each defect is shown in Table 2 (see Table 2).

Based on the criticism of the requirement error taxonomy, requirement errors have been identified. The three high-level classes of requirements' errors are entirely consistent with the factors contributing to the formation of the SRS document. These high-level of requirements' errors are listed as follows.

- *People errors* include errors caused by people involved in the process of requirements document development (SRS authors).
- *Process errors* include errors caused by conducting the requirements engineering process incorrectly (the process of requirements analysis).

Table 1 Results of the comparative analysis of reading techniques.

Defect type	CBR	DBR	PBR
Ambiguous	9	11	15
Inconsistent	6	6	14
Incorrect	10	12	6
Omission	23	42	46
Superfluous	2	2	0
Total	50	73	81

Table 2 Types of defect definitions.

Defect	Description
Omission	Necessary information related to the problem being solved by the software has been omitted from requirements document or are not complete
Ambiguous	The information written in the requirements document has more than one interpretation
Inconsistent	One part of the requirements document is inconsistent with other part/s or with the problem that the SRS solves
Superfluous	Some information of the SRS document is not relevant to the problem being solved or will not contribute to the solution
Incorrect	Some information in the SRS contradicts other information about the same or relevant information in the domain knowledge or conflict with preceding documents
Not-conforming to standards	Some items in the requirement are written in a way not conforming to the standards determined by quality assurance representatives
Not-implementable	Some requirements are not implementable due to system constraints, human resources, budget, or technology limitations
Risk-prone	Some requirements are risk prone due to unstable requirements or requirements with high interdependence

- *Documentation errors* include errors that occur during the process of writing the requirements regardless of the correctness of the requirements engineering processes or whether the author understood the requirements or not.

The three high-levels of requirements' errors will be used in the taxonomy. Some of the detailed errors which are embedded within each high-level class of requirements' errors will be excluded and others will be renamed, as follows.

Included errors are the errors that can be used in defect detection, as listed below.

- *Communication errors* include errors resulting from miscommunications among various stakeholders involved in authoring the requirements document.
- *Participation errors* are errors that result from inadequate or missing participation of important stakeholders involved in developing the requirements document.
- *Domain knowledge errors* occur when the requirements' authors lack knowledge or experience about the problem domain.
- *Specific application knowledge errors* occur when the requirements' authors lack knowledge about some aspects about the application or problem domain.
- *Elicitation errors* are produced from the use of an inadequate requirement elicitation process.
- *Analysis errors* include errors committed during the requirements analysis process.
- *Traceability errors* result from an inadequate or incomplete requirement traceability process to predecessors and successors, and the change management process.
- *Organization errors* arise during listing and organizing the requirements illogically and ineffectively during the documentation process.
- *No-usage of documentation standard errors* result from writing requirements in a way that is not conforming to the documentation standards that have been determined by the quality assurance representative.
- *Specification errors* occur during describing the requirements regardless of whether the developers correctly understood the requirements or not. Examples include typo errors, repetition of some steps, and mistakes in naming and referencing requirements.

Excluded errors are the errors that cannot be used to explain the reasons of defect occurrence, as listed below.

- *Process execution errors* include errors that occur when requirement authors make mistakes while executing the requirement elicitation and development processes regardless of the adequacy of the chosen process (as defined by Walia) [7]. However, the reviewer cannot identify the process execution error without asking the author of the specific part of the requirements that contains defects about the reason for the defect. Executing the requirement elicitation and development processes is a long trip. As the inspector is not often the author of the SRS, it is difficult to determine that the cause of the defect is a process execution error.
- *Inadequate method of achieving goals and objective errors* In this class of errors, Walia included errors that result from selecting inadequate or incorrect methods for achieving the stated goals and objectives of the SRS. For example, he included the following errors.
  - "System-specific information has been misunderstood leading to the selection of a wrong method." This type of errors is more suitable to be included in the *specific application knowledge errors* class.
  - "Selection of methods those were successful in other projects without investigating whether these methods are proper for this project." This type of error is more suitable to be included in the *elicitation errors* or *analysis errors* class according to the state of the real defect situation.

This class will be excluded because it is not precisely defined and the contained errors could be classified in other categories.

- *Management errors include errors* that result from inadequate or poor management processes [7]. Walia put these examples to explain this class of errors: misunderstandings about the assignment of resources to different development tasks, lack of leadership and misunderstanding of all the alternatives [29]. All these mistakes can be classified in other classes as *elicitation errors* or *analysis errors* class. It is not reasonable to determine the presence of the managerial errors or a lack of leadership errors without asking the author of the part that contains the defect about the reasons for his mistake. Often, despite the existence of managerial problems, the author may or may not commit a defect. Simply, there is no correlation.

Renamed errors are errors that have been renamed to become more expressive of their content. Renamed errors are explained as follows.

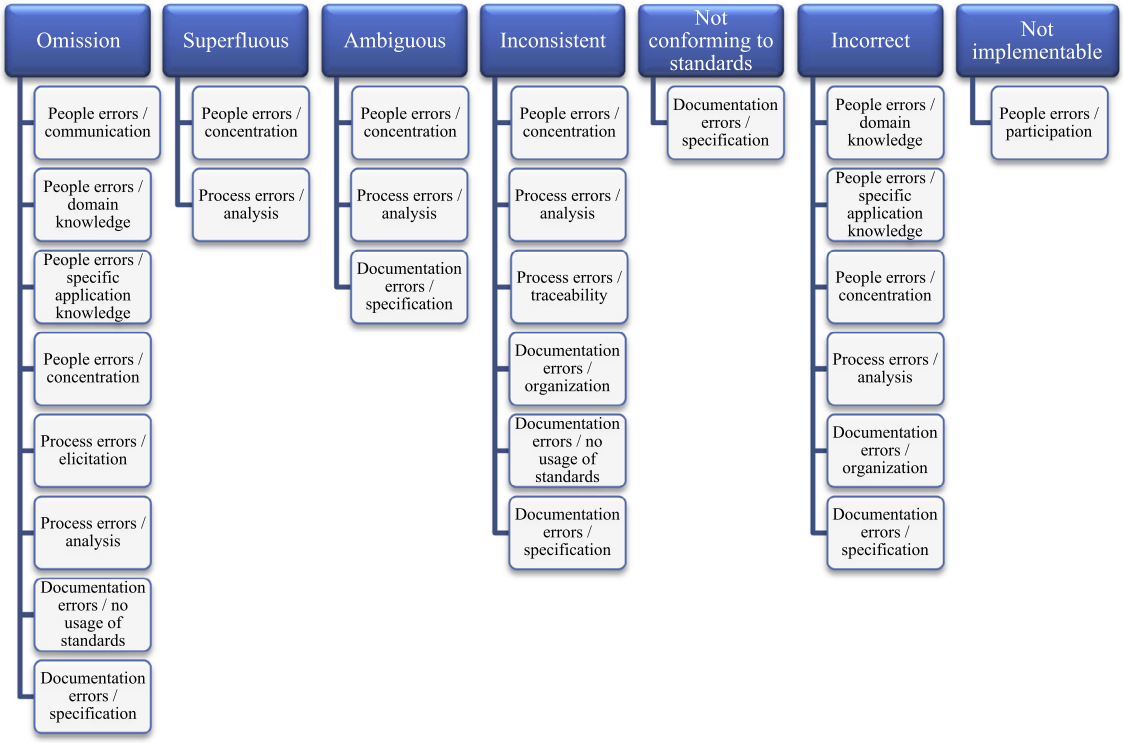

Figure 2 The taxonomy of defect types and their sources.

- *Other cognitive errors* This includes – as Walia [7] defined – errors that result from constraints on the cognitive abilities of the requirement authors. He also explained that other cognitive errors include errors caused by adverse mental states, mental fatigue and lack of motivation, and errors caused by the impact of environmental conditions on requirements author [29]. All such errors affect the concentration of the requirement's author. Therefore, these mistakes can be called *concentration errors* and remain embedded in people errors class.

Based on the previous analysis, the classes of errors that will be used in the taxonomy to determine the causes of defects are: communication errors, participation errors, domain knowledge errors, specific application knowledge errors, concentration errors, elicitation errors, analysis errors, traceability errors, organization errors, no-usage of documentation standards errors and specification errors.

To determine the relationship between each defect and the errors that caused it, an experiment has been conducted on three SRS documents. The three case-studies are listed below.

- Automated Teller Machine network (ATM) [23].
- The Online National Election Voting System (ONEV) [24].
- Web publishing system [25].

The experiment conducted by detecting defects on the three case-studies by using three reading techniques which are: CBR, PBR and DBR. The discovered defects were classified based on the causes of its occurrence by using the three high-level classes of requirements' errors (people errors, process errors and documentation errors) and then classified further to the detailed error classes as shown in Fig. 2.

6. The proposed combined-reading technique

This section presents the proposed combined-reading technique to detect defects in requirements phase in SRS documents. In Section 6.1 the need for the proposed technique is specified. The basis upon which the combined-reading technique is built is analyzed in Section 6.2. The combined-reading technique is proposed in Section 6.3.

6.1. The necessity of a new technique

The number and types of detected-defects vary from one technique to the other. To find out the reason behind that, it is essential to study the nature of each technique separately, the nature of each defect type, its source of occurrence and the ability of each defect detection technique to detect the defect. According to the comparative analysis of reading techniques presented in section four and by studying the questions of the checklists in each technique, the following results were found.

- In the Checklist-Based-Reading (CBR) technique, the focus of the questions is on covering the different parts that constitute the SRS document, but the questions do not cover all types of defects. The questions omitted many potential types of defects in each part of the SRS document.
- The Defect-Based-Reading (DBR) technique focuses on different types of defects, but it is not structured in terms of the different parts that constitute the SRS document. The DBR questions focus on each type of defects in some parts of the SRS document but not in every part of the SRS document. Also, the DBR technique did not focus on the various reasons for the emergence of each type of defects.

- The Perspective-Based-Reading (PBR) technique focuses mainly on the use-cases from the different perspectives of different stakeholders of the SRS document. The PBR technique did not take into account the components of the requirements document and the relationship of each part with others.
- As for the ambiguous and omission defects, the PBR detects a larger number of defects compared to DBR and CBR. We conclude that, the PBR is the most effective while DBR is more effective than CBR.
- Regarding the inconsistent defects, the PBR is more effective than DBR and CBR whereas both CBR and DBR are equal in the number of detected defects.
- Concerning the incorrect defects, the DBR is more effective than CBR and PBR while CBR is more effective than PBR.
- With regard to the superfluous defects, the CBR and DBR are equal in their effectiveness. While the PBR could not find any superfluous defects.
- All techniques do not give enough attention to the recommendations of the IEEE STD 830-1998 [30] that describe the content and quality of a proper software requirements specification (SRS) document.

It can be concluded that it is difficult to detect defects by using a single reading technique. Therefore, there is a need for a new technique to detect defects in requirements documents taking into account the relationship between defect types and their sources. The new technique should combine the advantages of these reading techniques and avoid their limitations. Thereby, the new technique contributes to the creation of a more efficient analysis compared to existing reading techniques; which should be reflected in the quality of the SRS documents.

6.2. The basis upon which the combined-reading technique is built

This reading technique is built to check the SRS content area for defects that can occur. So, the specific checkpoints are defects. However, not all defects could occur in all parts of the SRS document. Each part of the SRS document has a set of defects that can appear in it particularly. Therefore, the checklist will verify the existence of these defects. Each part has a purpose that must be studied to determine the types of defects that are likely to appear and that affect the achievement of the goal. Therefore, when checking the absence of these defects, the goal of each part of the SRS document will be verified. To do so, the following steps have been followed.

1. Each component of the document has been analyzed to its basic elements.
2. Based on the previous knowledge of defects as studied in the taxonomy of requirements' defects in section five, for each element, the different types of potential defects and their reasons for occurrence were determined.
3. For each element, questions were developed to cover the different types of potential defects from the different stakeholders' perspectives whenever possible.
4. Dividing questions into groups, based on the essential parts of the SRS recommendation of the IEEE STD 830-1998 [30]. Taking into account that each set of questions covers an essential part of the major components of the SRS document.

6.3. The combined-reading technique

The main idea behind the *combined-reading technique* is to examine every part of the SRS document separately. In each part, a search is performed for the expected sources of defects based on the previous knowledge of defects in a particular part and the purpose of each part of the SRS document. The technique is expressed in the form of questions in order to investigate special kinds of defects. Inspectors read the document during answering a list of yes/no questions that are based on the previous knowledge of frequently occurring defects. This technique aims to serve untrained inspectors. Every question in the method inspects a different part of the SRS document for the type of defect and its cause. The technique is presented in Appendix a and is explained in details in [31].

7. Evaluation

The aim of the evaluation is to verify the taxonomy of defects in requirements specification, to support inspectors to choose the suitable reading technique, and to investigate to what extent the combined-reading technique does improve defect detection in SRS documents.

7.1. Experiment

To perform this evaluation, an experiment has been conducted on four case-studies described as follows.

- The ***Parking Garage Control System* (*PGCS*)** [32] is a system that controls and supervises the entries and exits of a parking. The total number of defects found during the inspection experiment is 51 defects. This SRS has been used many times previously by University of Maryland and the International Software Engineering Research Network (ISERN) to conduct experiments to evaluate reading techniques.
- The ***ABC Video System*** [33] is a system that automates and reports the video rental and return processes. The total number of defects found in that SRS during the inspection experiment is 84 defects. This SRS has been used previously by University of Maryland and the ISERN to conduct experiments to evaluate reading techniques.
- The ***Online Shopping Mall* (*OSM*)** [34] application is an online shopping site to enable vendors to set up online shops, help customers to browse through the shops, purchase online without the need to visit the shop physically and give feedback for the received product or service. In addition, the application provides a system administrator the ability to approve and reject requests for new shops and maintain lists of shop categories. The total number of defects found in that SRS during the inspection experiment is 238 defects.
- The ***Massively Multiplayer Online Role Playing Game* (MMORPG)** reported in [35] is an online game to promote cultural tourism in Turkey. The player travels over the Turkish cities in order to perform missions and collect coins and the gold scattered over some secret places. The total number of defects found in that SRS during the inspection experiment is 44 defects.

The experiment was performed by applying the defect detection techniques (reading techniques) on each of the four SRSs case-studies. The detected-defects are classified by the type of defect, the main-reason for their appearances (the three high-level classes of requirements errors) and classified further to the sub-reason for their appearance.

7.2. Results

The experiments' results gathered from the four SRSs used for the evaluation purpose are presented in Table 3.

7.3. Analysis of results

In the following subsections, the results of the evaluation experiment are analyzed.

7.3.1. Evaluating the taxonomy of defects

The taxonomy distinguishes between eight types of defects, which are *omission, ambiguous, inconsistent, superfluous, incorrect, not conforming to standards, not implementable*, and *risk-prone*. The labeling clearly describes each of them. The taxonomy is hierarchical and contains three levels. This structure distinguishes between the type of defect, the main-source of its occurrence and the sub-source for the occurrence. This is in order to determine the radical causes of each type of defects.

To verify the taxonomy, we inspected defects in these four SRS documents by four different reading techniques, which are CBR, PBR, DBR and combined-reading in order to detect the largest possible number of defects. The total number of distinct defects found in the SRSs during the inspection experiment is 417 defects classified into six defect types (*ambiguous, incorrect, inconsistent, omission, superfluous* and *not conforming to standards*). This is consistent with the proposed taxonomy of defect types and their sources. This confirms further the stability and reliability of the taxonomy. However, we encourage other researchers to do other experiments to verify the stability and reliability of the taxonomy and to extend the taxonomy if there were other sources for defect occurrence.

Compared to previous classifications, the proposed taxonomy differs in the points stated as follows.

- The objective of the IEEE categorization of anomaly classes is to classify types of defects in (SDLC) to improve the quality of software. While the objective of the classification of Margarido et al. is to classify defect types in SRS documents in order to build checklists to support requirements reviewers, the objective of Walia's classification is to classify errors that cause defects in the requirements phase generally to support the prevention and detection of errors while the objective of the proposed taxonomy is to classify defects and their causes in requirements phase (in SRS documents) to specify the correlation between the defects and their sources in order to guarantee the quality of the SRS document.
- An empirical research approach has been used to create the classification of Margarido et al. for defects, the classification of Walia et al. for requirements' errors and the proposed taxonomy for defects and their sources of occurrence, while the methodology used to conduct the IEEE categorization of anomaly classes is not clearly stated.
- The IEEE categorization does not focus on any phase or any type of document. It works on all (SDLC) phases. While the classification of Margarido et al. is concerned with the requirements phase and specifically for the SRS documents, the Walia's classification of requirements' errors did not focus on any type of requirements documents while the proposed taxonomy is concerned with requirements phase and specifically the SRS documents.
- The IEEE categorization of anomaly classes did not define or mention any characteristics of these anomalies. The IEEE categorization does not specify when and why an anomaly occurs or how to detect its existence, while the classification of Margarido et al. of defect types defined each defect type and gave an example on each of them but they did not specify the causes of these defects. Also, Walia's classification of requirements' errors defined each requirement error and gave an example on each of error, but they did not relate the types of defects and errors leading to their appearance. While the proposed taxonomy defined each defect, specified the causes of these defects and gave an example on each pair of defect and error causing it.
- The root-cause-analysis method used for problem solving in the classification of Margarido et al. for defects, the classification of Walia et al. for requirements' errors and the proposed taxonomy for defects and their sources of occurrence; taking into account the difference in research objective and problem. With regard to the IEEE categorization of anomaly classes, the used problem solving method is not clear.

Table 3 Results of experiment used for evaluating the combined-reading technique.

Defect type	CBR	DBR	PBR	Combined-reading
Ambiguous	23	11	9	46
Inconsistent	1	1	4	8
Incorrect	16	17	9	30
Not conforming to standards	2	2	1	1
Omission	92	123	169	196
Superfluous	8	7	5	18
Total	142	161	197	299

7.3.2. Evaluating the combined-reading technique

In order to evaluate the combined-reading technique, its effectiveness in detecting the number and variety of defects is compared with the effectiveness of other defect detection techniques. The detected-defects by each of the techniques have been compared with the detected-defects by other techniques. The comparison has been performed on the following aspects: the total number of detected-defects, the correlation between defects and source of occurrence, and both the common and different detected-defects between other techniques.

An experiment has been conducted on the same four SRS documents to evaluate the combined-reading technique. We inspected defects in these four SRS documents by four different reading techniques, which are CBR, PBR, DBR and combined-reading, in order to compare the effectiveness of these techniques. From the experiments that we performed, we noted that the combined-reading technique is the most efficient

in detecting defects in terms of the number of detected-defects and the variety of types of detected-defects followed by the Perspective-Based-Reading (PBR). The Defect-Based-Reading (DBR) is in the third rank in terms of the number of detected-defects. Checklist-Based-Reading (CBR) is ranked in the fourth rank, as shown in Table 3.

To determine the efficiency of each technique in detecting each type of defects, we examined the areas of differences and commonalities among the given techniques. Concerning the *ambiguous* defect type, the combined-reading is absolutely the best, and far ahead from the rest of the techniques. The CBR follows the combined-reading technique. The CBR shares the same number of results with the combined-reading technique in detecting sixteen ambiguous defects and exclusively detects seven defects. In the third rank is the DBR and in the fourth the PBR.

The *inconsistent* defect type has been detected by the combined-reading technique and the PBR respectively. The CBR and the DBR are equal in detecting the *inconsistent* defect. In terms of the number of detected-defects, the combined-reading technique is better than the PBR techniques. However, each of them detected different defects.

Regarding the *incorrect* defect, the combined-reading technique is far ahead of the rest of the techniques. The CBR and the DBR are almost equal in efficiency of detecting the *incorrect* defects. The PBR technique is the worst in detecting the *incorrect* defects.

The combined-reading technique is the best in detecting the *omission* defects. The PBR technique comes in the second rank. The combined-reading technique and the PBR technique shared detecting ninety-seven *omission* defects. The DBR technique comes in the third rank and in the fourth rank comes the CBR technique.

With respect to the *superfluous* defects, the combined-reading technique is the best in detecting such type of errors. The other three reading techniques are almost equal in detecting the *superfluous* defects. Regarding the detection of the *not conforming to standards* the four reading techniques are almost equal.

Based on the foregoing, the combined-reading technique is better than other techniques, followed by the PBR technique. Then, in the third rank comes the CBR technique and in the fourth rank the DBR technique comes.

7.3.3. Advantages of the combined-reading technique

- It guides the inspector about what is being inspected and provides instructions on how the inspection can be performed.
- The technique does not need much training to use it.
- It is better for reviewers who are less familiar with the software domain.
- The technique detects a large number of defects and a variety of defect types compared to other reading techniques (CBR, DBR and PBR).
- Questions that constitute the combined-reading technique are based on the errors causing defects. Thus, the combined-reading technique guides inspectors to the reasons of defect occurrence. Therefore, the technique does not rely on inspector's guessing to determine the cause of the defect.

7.3.4. Disadvantages of the combined-reading technique

- Questions are limited to the detection of defects that have been previously detected throughout the study. Therefore, inspectors may not focus on defect types that have not been previously detected.
- It does not provide instructions on how to divide the inspection process between inspectors in order to determine the responsibility of each of them.
- It is a non-automated technique.

Despite these shortcomings, the most of them can be overcome by automating the combined-reading technique, and further study of the sources of defects.

8. Conclusion and future work

Existing taxonomies of defects in the requirements phase classify defects regardless of their causes. While Walia's taxonomy classified the causes leading to defects in the requirements phase, however, it ignored the relationship between defects and errors that cause it. The proposed taxonomy focused on the defects in the requirements phase and added correlation between the defects and its sources to guarantee the quality of the SRS document. The taxonomy helps in training the SRS writers, and supports creating more accurate and efficient defect prevention and defect detection techniques. In addition, this work presented the Combined-Reading technique to detect defects in the software requirements specification (SRS) documents. The Combined-Reading technique combines the advantages of other reading techniques in defect detection.

The taxonomy of requirements defects needs to be validated further on other experiments to determine its stability and reliability. There is a need for developing automated defect detection techniques to reduce the reliance on the human factor that depends on cognitive abilities. Also, the Combined-reading technique needs more studies and experiments to provide confirmation of its effectiveness on SRSs from different domains and with different sizes. A formal way for computing the complexity should also be investigated.

Despite the exerted efforts in the research of the requirements inspection, many open questions that are still not answered. Such questions include the following.

- What is the most efficient inspection-based technique?
- What is the suitable requirements defect detection technique for each SDLC model?
- Is the requirements inspection a group work or an individual work?
- Does the requirements validation-based technique differ from one software domain to the other?

Appendix A

The combined-reading technique.

Q.	Checked item	Defect type	Defect main-source	Defect sub-source	Check-list items
1	External interfaces	Omission	Process errors	Analysis	Have all inputs and outputs of the software system been described in details? (oriented to the developer)
2	User interfaces	Omission	People errors	Concentration	Have all interfaces between the software product and its users been specified? (oriented to the end-user)
3	Constraints	Omission	Process errors	Analysis	Are all significant consumers have scarce resources; as memory, network bandwidth, processor capacity... identified? Has the anticipated consumption of resources been specified?
4	Use-case name and number	Ambiguous	People errors	Concentration	Does the use-case name reflect its goal?
5		Inconsistent	Documentation errors	Organization (organizing the SRS document)	Are the requirements arranged numerically according to the logical order of occurrence and in order to prevent confusion between the requirements?
6		Inconsistent	Process errors	Traceability	Is there a conflict between the functional requirements names and the relevant use-cases names?
7	Actor	Ambiguous	Documentation errors	Specification (stating the requirements)	Are actors clearly specified for each functional requirement?
8		Incorrect	People errors	Concentration	Do you agree that the assigned actor is suitable for the connected use-case?
9		Omission	People errors	Concentration	Are there any missing actors (system, hardware, or human user), although it is known that this actor should be included in this functional requirement?
10		Omission	Process errors	Analysis	Are there any missing actors (system, hardware, or human user)?
11	Description	Ambiguous	Documentation errors	Specification (stating the requirements)	Is the description of the functional requirements written in a clear and concise language?
12		Inconsistent	People errors	Concentration	Are the descriptions of the functional requirement consistent with what is specified in the product functions section in terms of actor, flow of events?
13	Pre-condition	Ambiguous	Process errors	Analysis	Do you think that the pre-conditions are described clearly?
14		Omission	Process errors	Analysis	Is there a missing pre-condition for any use-case that should be existed?
15		Incorrect	Process errors	Analysis	Is the pre-condition correctly defined and consistent with the goal of the functional requirement and with you domain knowledge?
16	Input	Ambiguous	Documentation errors	Specification (stating the requirements)	Is the input written in a clear and concise language, and is linked to the output of the previous functional requirement (if any)?
17		Incorrect	People errors	Concentration	Is the expected input correctly defined and consistent with what is written in the description section of the functional requirement?
18		Omission	Documentation errors	Specification (stating the requirements)	Are there any omitted parts of the details of the inputs of the functional requirements, including their source, accuracy, range of values, frequency, units and format?
19		Omission	People errors	Concentration	Are there any missing inputs to the functional requirements; although it has been mentioned elsewhere in the document?
20		Omission	Process errors	Analysis	Are all the inputs of the software specified?
21		Inconsistent	Process errors	Analysis	Does the data input of the functional requirement conflict with the logical structure of the database in the non-functional requirements section?

(*continued on next page*)

Q.	Checked item	Defect type	Defect main-source	Defect sub-source	Check-list items
22	Trigger	Inconsistent	Documentation errors	Specification (stating the requirements)	Is it possible that conflict occurs as a result of the way of writing the trigger of the functional requirement and the way of writing the trigger of the use-case?
23		Inconsistent	Process errors	Analysis	Do you think that there is conflict between the trigger of the functional requirement and the trigger section of its use-case?
24	Interactions (Normal flow of events)	Ambiguous	Documentation errors	Specification (stating the requirements)	Is the flow of events written clearly and unambiguously? In order to assist the analyst and the designer to make sure of what they should do
25		Ambiguous	Process errors	Analysis	May the flow of events give different outputs according to the interpretation of a requirement or a functional specification?
26		Incorrect	Documentation errors	Specification (stating the requirements)	Is the sequence of functions in the functional requirement written in a clear and correct way that reflects the intended goal of the function? Do you think that the tasks provided by the use-case written in a clear and correct way to reflect the intended goal?
27		Incorrect	People errors	Domain knowledge	Is the goal of the sequence of functions or tasks in the normal flow of events that the functional requirement provides, fulfilled?
28		Incorrect	Process errors	Analysis	Does the sequence of functions or tasks in the normal flow of events, provided by the functional requirement, fulfill the goal of it?
29		Inconsistent	Process errors	Analysis	Does the behavior of the functional requirement conflict with the behavior of other use-cases? Does the behavior of the functional requirement conflict with your domain knowledge or the software general description?
30		Inconsistent	Process errors	Analysis	Is there a conflict between the behavior of the functional requirement and the behavior of its use-cases?
31		Omission	People errors	Concentration	Is there any missing functionality or task that should be provided by the use-case to fulfill its goal, although this function has been mentioned elsewhere in the document?
32		Omission	Process errors	Analysis	Is there any missing functionality or task that should be provided by the use-case to fulfill its goal? Is the flow of events described with concrete terms and measurable concepts? Does the flow of events need to be described with more details for design?
33		Superfluous	Process errors	Analysis	Is there any superfluous functionality or task that will not contribute to achieve the goal of the use-case? Is there any superfluous functionality or task that is not programmable, in the flow of events of the functional requirements?
34	Exceptional flow of events	Ambiguous	Documentation errors	Specification	Is the exceptional flow of events written clearly and unambiguously? In order to assist the analyst and the designer to make sure of what they should do.
35		Incorrect	Process errors	Analysis	Based on your domain knowledge, are the described variants to the normal flow of events correct and in a way that fulfills their goal? Do they make sense?
36		Omission	Process errors	Analysis	Are there any missing variations of the normal flow of events that have not been identified in the functional requirements?
37		Superfluous	Process errors	Analysis	Is there any superfluous functionality or task that will not contribute to achieve the goal of the functional requirements?
38	Post-conditions	Ambiguous	Process errors	Analysis	Are described post-conditions clear?
39		Incorrect	Process errors	Analysis	Is the expected post-conditions correctly defined? Does it lead to achieve the goal of the functional requirement?
40		Omission	Process errors	Analysis	Are the post-conditions described for all functional requirements? In order to indicate the achievement of the goal of the functional requirement

Q.	Checked item	Defect type	Defect main-source	Defect sub-source	Check-list items
41	Output	Ambiguous	Documentation errors	Specification (stating the requirements)	Is the output written in a clear and concise language that informs the designer what should be done and describes the outputs to the end user?
42		Ambiguous	Process errors	Analysis	Does the output reflect the success or failure of the functional requirements?
43		Incorrect	Process errors	Analysis	Is the expected output correctly defined? Does it lead to achieve the goal of the functional requirement?
44		Omission	Documentation errors	Specification (stating the requirements)	Are there any omitted parts of the details of the outputs from the functional requirements that have been specified for both normal flow of events and variations, including their destination, accuracy, range of values, frequency, and format?
45		Omission	People errors	Concentration	Are there any missing outputs of the functional requirements for both normal flow of events and variations, although it has been mentioned elsewhere in the document?
46		Omission	Process errors	Analysis	Have all the outputs of the functional requirements been specified for both normal flow of events and variations, including their destination, accuracy, range of values, frequency and format?
47		Inconsistent	Process errors	Analysis	Does the output of any functional requirement facilitate referencing to the input or the pre-condition of the following functional requirement?
48	Criticality	Omission	Documentation errors	No use of documentation standards	Is the criticality of functional requirements identified?
49	Priority	Omission	Documentation errors	No use of documentation standards	Is the implementation priority of each functional requirement identified? Are the criteria for assigning functional requirement priority levels been defined?
50	Activity diagram	Ambiguous	Documentation errors	Specification (stating the requirements)	Is every component in the activity diagram written in a way reflects what it actually does and to enhance understandability of the requirements? Does each activity diagram have name?
51		Incorrect	People errors	Concentration	Is the name of activity diagram correct, in the sense that it reflects the goal of it?
52		Inconsistent	Documentation errors	Specification (stating the requirements)	Is there a conflict between the components of the activity diagram, or between them and the textual descriptions of the functional requirement?
53		Superfluous	Process errors	Analysis	Is there any superfluous activity in the activity diagram? Is the activity diagram too detailed to restrict the design?
54	Traceability	Omission	Documentation errors	Specification (stating the requirements)	Have all internal cross-references to other requirements been specified?
55	Overall use-case and its relation with other use-cases	Ambiguous	Documentation errors	Specification (stating the requirements)	Is every requirement written in a clear and concise language to ensure a unique interpretation?
56		Ambiguous	Process errors	Analysis	Has every requirement been specified separately and been avoiding compound requirements?
57		Incorrect	Documentation errors	Organization (organizing the SRS document)	Are all functional requirements actually functional requirements and not constraint?
58		Incorrect	People errors	Concentration	Does the functional requirement lead to fulfill the goal of the application? Is the functional requirement consistent with the sections: purpose and scope of the SRS document?
59		Incorrect	Process errors	Analysis	Are all functional requirements actually functional requirements?

(*continued on next page*)

Q.	Checked item	Defect type	Defect main-source	Defect sub-source	Check-list items
60		Inconsistent	People errors	Concentration	Is there any typo error in the requirement that may lead to contradictory results and inconsistent requirements especially in parameters?
61		Omission	Process errors	Analysis	Have the time criteria been specified for the functional requirements?
62		Omission	Process errors	Analysis	Are there any missing functional requirements that may contribute to achieve the goal of the program? If so, is it identified as TBD? Do the requirements provide an adequate basis for design?
63		Omission	Process errors	Analysis	According to your domain knowledge or the introduction section, has the SRS omitted functional requirements that you think they are necessary?
64		Superfluous	Process errors	Analysis	Is there any superfluous use-case that will not contribute to achieve the goal of the program? Is there any superfluous use-case that is not mentioned in the general description of the software product?
65	Performance requirements	Incorrect	Documentation errors	Organization (organizing the SRS document)	Is there any confusion between performance requirements and design constraints or any other constraints?
66		Omission	Process errors	Analysis	Are both static- and dynamic- numerical performance requirements specified precisely?
67		Omission	Process errors	Elicitation	Are both static- and dynamic- numerical performance requirements specified?
68	Non-functional requirements	Omission	Process errors	Elicitation	Are the necessary non-functional requirements specified, including reliability, availability, security, maintainability and portability?
69	General questions	Inconsistent	Documentation errors	No use of documentation standards	Did the author of the SRS document use numbering levels to identify sections of the SRS and the functional requirements?

References

[1] A. Aurum, C. Wohlin, Engineering and Managing Software Requirements, Springer-Verlag New York Inc., Secaucus, NJ, USA, 2005.

[2] W.M. Wilson, Writing effective natural language requirements specifications, Crosstalk J. Def. Softw. Eng. (1999) 16–19.

[3] Effective Requirements Definition and Management: Improves Systems and Communication (White paper), Ed. 8310 North Capital of Texas Highway, Corporate Office: Whitepaper of Borland Software Corporation - The open ALM Company, 2009.

[4] R. Chughtai, A Domain-Specific approach to Verification & Validation of Software Requirements, M.Sc. thesis, Arizona State University, 2012.

[5] IEEE, IEEE Standard for Software Reviews, IEEE Std 1028–1997, 1998, pp. i–37.

[6] I.L. Margarido, J.P. Faria, R.M. Vidal, M. Vieira, Classification of defect types in requirements specifications: Literature review, proposal and assessment, Presented at the 2011 6th Iberian Conference on Information Systems and Technologies (CISTI), Chaves / Portugal, 2011.

[7] G.S. Walia, J.C. Carver, A systematic literature review to identify and classify software requirement errors, Inform. Softw. Technol. 51 (2009) 1087–1109.

[8] L. He, J. Carver, PBR vs. checklist: a replication in the n-fold inspection context, Presented at the Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering (ISESE'06), Rio de Janeiro, Brazil, 2006.

[9] G. Sabaliauskaite, F. Matsukawa, S. Kusumoto, K. Inoue, An Experimental Comparison of Checklist-Based Reading and Perspective-Based Reading for UML Design Document Inspection, Presented at the Proceedings of the 2002 International Symposium on Empirical Software Engineering, 2002.

[10] F. Lanubile, G. Visaggio, Evaluating defect detection techniques for software requirements inspections, International Software Engineering Research Network (ISERN) Report no00-08, 2000.

[11] M. Ciolkowski, What do we know about perspective-based reading? An approach for quantitative aggregation in software engineering, Presented at the Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, 2009.

[12] A. Porter, L. Votta, Comparing detection methods for software requirements inspections: a replication using professional subjects, Empirical Softw. Eng. 3 (1998) 355–379.

[13] A.A. Porter, L.G. Votta, An experiment to assess different defect detection methods for software requirements inspections, Presented at the Proceedings of the 16th international conference on Software engineering, Sorrento, Italy, 1994.

[14] A.A. Porter, J. Lawrence, G. Votta, V.R. Basili, Comparing detection methods for software requirements inspections: a replicated experiment, IEEE Trans. Softw. Eng. 21 (June 1995) 563–575.

[15] T. Thelin, P. Runeson, C. Wohlin, An experimental comparison of usage-based and checklist-based reading, IEEE Trans Softw. Eng. 29 (2003) 687–704.

[16] L.H. Sulehri, Comparative Selection of Requirements Validation Techniques Based on Industrial Survey, M.Sc. Thesis, Department of Interaction and System Design, School of Engineering, Blekinge Institute of Technology, 2009.

[17] S.B. Saqi, S. Ahmed, Requirements Validation Techniques Practiced in Industry: Studies of Six Companies, M.Sc. thesis, Department of System and Software Engineering, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden, 2008.

[18] M. Staron, L. Kuzniarz, C. Thurn, An empirical assessment of using stereotypes to improve reading techniques in software inspections, Presented at the Proceedings of the Third Workshop on Software Quality (WoSQ '05), St. Louis, Missouri, 2005.

[19] A. Aurum, H. Petersson, C. Wohlin, State-of-the-art: software inspections after 25 years, Softw. Test. Verif. Reliab. 12 (2002) 133–154.

[20] O. Laitenberger, A survey of software inspection technologies, Handbook Softw. Eng. Knowl. Eng. 2 (2002) 517–555.

[21] X.M. Yang, Towards a Self-evolving Software Defect Detection Process, M.Sc. Thesis, Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, 2007.

[22] G.H. Travassos, F. Shull, M. Fredericks, V.R. Basili, Detecting defects in object-oriented designs: using reading techniques to increase software quality, Presented at the Proceedings of the 14th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA), Denver, Colorado, United States, 1999.

[23] J. Raymond, Schneider, Requirements Document for an Automated Teller Machine Network (ATM), Private Communication, 1996, pp. 1-17.

[24] E. Joldoshev, H.S. Matar, M.B. Özkan, H. Lutin, Software Requirement Specification For Online National Election Voting System, Computer Engineering Department, Middle East Technical University, 2010, pp. 1–55.

[25] J. Teamleader, P. Adams, B. Baker, C. Charlie, Web Publishing System, Michigan State University, 2004, pp. 1–31.

[26] G. Travassos, F. Shull, M. Fredericks, V.R. Basili, Detecting defects in object-oriented designs: using reading techniques to increase software quality, Presented at the Proceedings of the 14th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, Denver, Colorado, United States, 1999.

[27] C. Denger, T. Olsson, Quality Assurance in Requirements Engineering, in: A. Aurum, C. Wohlin (Eds.), Engineering and Managing Software Requirements, Springer, Berlin, Heidelberg, 2005, pp. 163–185.

[28] B. Anda, D.I.K. Sjøberg, Towards an inspection technique for use case models, Presented at the Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02), Ischia, Italy, 2002.

[29] G.S. Walia, J.C. Carver, A Systematic Literature Review to Identify and Classify Software Requirements Errors, University of Alabama MSU-071207, 2007.

[30] IEEE, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. 830–1998 (Revision of IEEE Std. 830–1993), 1994, pp. 1–40.

[31] A. Alshazly, A Combined Reading Technique to Detect Software Defects in Requirements Phase, M.Sc. thesis, Department of Information Technology, Institute of Graduate Studies and Research (IGSR), University of Alexandria, Alexandria, Egypt, 2013.

[32] R.J. Schneider, Requirements document for a parking garage control system (PGCS), Private Communication, 1996, pp. 1–18.

[33] R.J. Schneider, Requirements Document for ABC Video System, Private Communication, 1996, pp. 1–14.

[34] S. Ray, S. Bhattacharya, S. Shaw, S. Sett, Online Shopping Mall Project Report, Department of Information Technology, B.P. PODDAR Insititute of Management and Technology, West Bengal University of Technology, 2009, pp. 1–50.

[35] C. Kilcioglu, M. Degirmenci, U.C. Buyuksahin, Software Requirements Specifications: Massively Multiplayer Online Role Playing Game Project (MMORPG), Department of Computer Engineering, Middle East Technical University, 2010, pp. 1-32.