VOL. 14, NO. 17, SEPTEMBER 2019 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2019 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

A SURVEY OF SOFTWARE REQUIREMENTS SPECIFICATION AMBIGUITY

Ashok Kumar Gupta[1, 2], Aziz Deraman[2] and Shams Tabrez Siddiqui[1]
[1]Department of Computer Science, Jazan University, Jazan, Kingdom of Saudi Arabia
[2]Universiti Malaysia Terengganu, Kuala Terengganu, Malaysia
E-Mail: kgupta.ashok@gmail.com

ABSTRACT

The quality of Software Requirements Specification (SRS) is critical. A good quality SRS must be clear, correct, consistent, unambiguous, modifiable, verifiable and traceable. Ambiguity is difficult to tackle; therefore, requirements elicitation technique should be effective. Published material related to SRS issues discusses ambiguity as the most conversed problem. In this survey, our focus is on one of the major quality issues i.e. ambiguity. There are many solutions to resolve SRS ambiguity but, there is no prescribed classification of these solutions exists. The survey to provide a summary of the huge research contributions in the form of developments and new techniques, models, and methodologies that have been recommended to control the SRS ambiguity and magnify the benefits of addressing SRS ambiguity in development projects. To organize this enormous work by researchers, we try to provide the main concepts and associations that together represent the field of SRS ambiguity. The study is important for further assessments of possible solutions to SRS ambiguity for the improvement of SRS that helps researchers and experts to compare these techniques for better results.

Keywords: SRS ambiguity, ontology, NLP, UML, boilerplates, inspections, controlled language, SRS quality, ambiguity tools.

1. INTRODUCTION

Requirements Engineering (RE) process is a critical step, because SRS quality issues are significantly important, for different software projects domains. Sometimes SRS quality is directly rated as the main cause of the disasters in software development projects [1-3]. The final output of effective RE comes out to be a good quality SRS [3, 4]. It has been more than 20 years since Shull, Forrest J Basili, Victor R. developed the defect taxonomy related to SRS that consists of six types of problems missing, incorrect fact, inconsistent information, ambiguous, extraneous information and miscellaneous [5]. IEEE Standard 830-1998 [6] provides the Characteristics of a good SRS. The characteristics consist of unambiguity, correctness, modifiability, completeness, traceability, and ranking for importance, consistency, stability, and verifiability. However, complete, accurate and steady SRS requires extensive research to achieve the accuracy level.

An evident research problem in RE is resolving ambiguity, where ambiguity can be defined as "a statement having more than one meaning". It appears that no single broad, all-inclusive and exact definition of ambiguity written in the software engineering work. Every definition provides only some parts and portions of the complete definition by neglecting the rest of the definition. Altogether it forms a complete understanding of the current definition of ambiguity in Software Engineering.

The IEEE Recommended Practice for Software Requirements Specifications [6] says that "An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation." The problem with the IEEE definition is that there is no unambiguous specification simply because for any specification, there is always someone who understands it differently from someone else, just as there are no bug-free programs. According to Gause and Weinberg [7], there are two major sources of ambiguity, communication errors and missing information. Communication errors occur due to expression insufficiencies and lack of contextual information between the author and the reader. Missing information can be due to many reasons, for example, human factor, lack of observation, and generalize inaccurately.

On the other hand, till date most of the research work on SRS ambiguity has not been assimilated in an organized manner, hence making the work difficult to reconcile and evaluate for researchers and practitioners. To provide an organized and systematized view of the research in SRS ambiguity, this survey describes the current state of the art of research work available in the field of SRS ambiguity. The survey includes taxonomy of the core concepts and relationships that together embody the SRS ambiguity field. This taxonomy is organized around two primary dimensions, technical, and tools with which we try to portray SRS ambiguity. While these major dimensions are most suitable for the key areas of software development, we elicited from the literature individual sub-dimensions that are crucial for the work in the field of SRS ambiguity. This material may concrete the road for the use of the SRS ambiguity method in projects. Moreover, it provides a road-map in the form of a classification that helps researchers to focus on the best suited solutions available for a particular ambiguity.

2. RELATED WORK

To consolidate and organize the findings of SRS issues, due to the sheer volume of work already published, is difficult. Here we will discuss most relevant surveys available. Broadly their surveys can be summarized as follows.

Gernat [8] describes the various types of ambiguities. He classified these ambiguities into six broad

VOL. 14, NO. 17, SEPTEMBER 2019 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2019 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

categories. *Lexical* ambiguity, *Semantic* ambiguity, Structural ambiguities etc. Mich [9] defined weighted semantic ambiguities, the weighted semantic ambiguity that includes the frequencies that the different meaning of the word occurs.

Shah and Jinwala [10] provided a comprehensive survey on the techniques to resolve ambiguities in natural language software requirements. The research was focused to compare and contrast the established approaches to deal with ambiguities in natural language software requirements. The considered research presented a survey of the currently available automatic and semi-automatic tools for ambiguity resolution. We observed that the study doesn't include all possible solutions to resolve the ambiguity. It focuses on the tools used to resolve the ambiguity. This article reports on our understanding on the basis of literature survey to categorize the solutions and to check the number of solutions available in each category.

All of these surveys can enhance the knowledge of ambiguities and types of ambiguities by identifying factors that may influence SRS. However, none of them offers its findings from a more comprehensive approach. It becomes difficult for researchers to decide which ambiguity detection method or tools to choose, and which one will be effective in case a particular type of ambiguity appears in SRS.

We conducted search in various databases for source selection, and inclusion and exclusion criteria. The guidelines are based on Petersen *et al.* [11]. Many authors worked on the types of ambiguities and suggested types of ambiguities and plenty of proposed and recommended solutions to handle ambiguities. The objective of the study is to offer a summary of the SRS research on effective solutions in terms of frameworks, models methods, and techniques available to handle SRS ambiguity.

2.1 Selection of source

The objective of the source selection is to search relevant sources of literature for the study. The sources of the literature are online journals, databases, thesis, and books. At first step, we considered the databases *ACM, IEEE, Springer, Scopus* and *Science Direct* as proposed in [12]. The major keywords are mined from a collection of related papers known by the researchers as extremely appropriate. We made a comparison to check whether our findings include publications from these peer-reviewed papers [10, 13-16] or not. We searched the literature from the above-mentioned sources. Selection of appropriate literature is made according to the process defined in Figure-1. Finally, we combined the relevant literature retrieved from all sources to get the required set of literature that is being used to find a solution for the research problem.

3. TAXONOMY TO RESOLVE AMBIGUITY

On the basis of extracted literature, we derived the nomenclature to clear the fundamental ideas and relationships of SRS ambiguity. This taxonomy is prepared around two key dimensions - technical dimension and tools dimension as shown in Figure-2. With them, we characterized the details about SRS ambiguity. We used a proper selection method for each key dimension, in the form of a predefined goal to get relevant papers from the large set of literature. However, required technical and tools dimensions are not exclusive, these dimensions are linked to the various issues of SRS, here we will focus only on one issue i.e. ambiguity. We found literature specific to sub-dimensions that is significant and of good relevance in context to resolve the ambiguity.

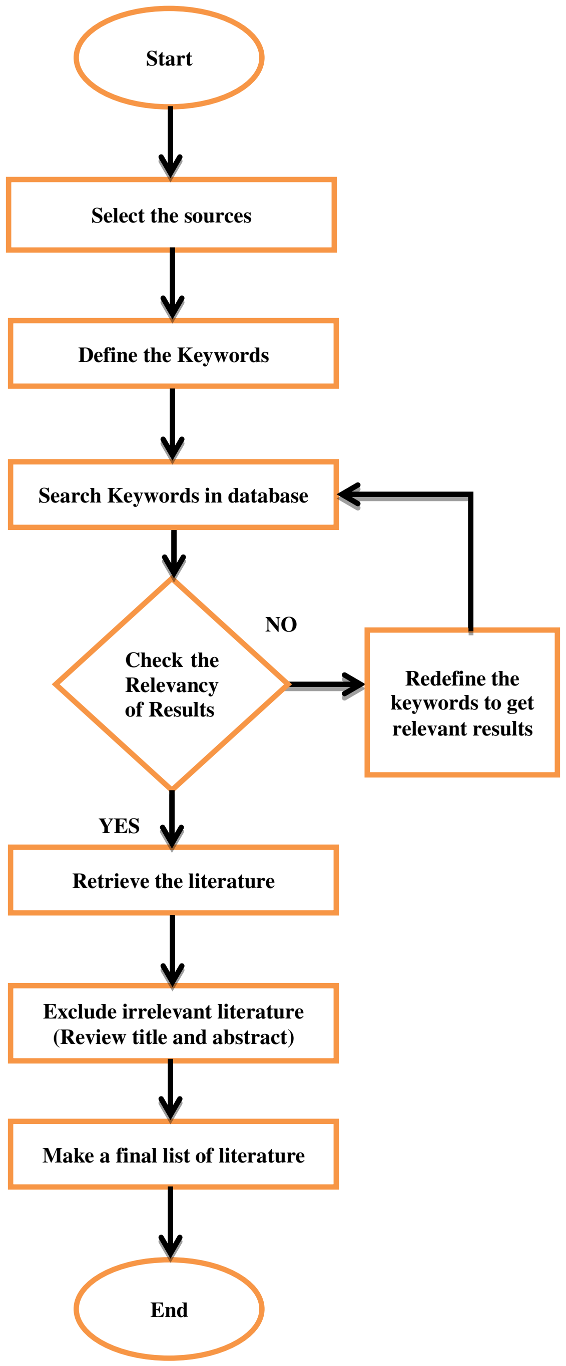
Start
Select the sources
Define the Keywords
Search Keywords in database
Check the Relevancy of Results
NO
Redefine the keywords to get relevant results
YES
Retrieve the literature
Exclude irrelevant literature (Review title and abstract)
Make a final list of literature
End

Figure-1. Process for literature search and selection.

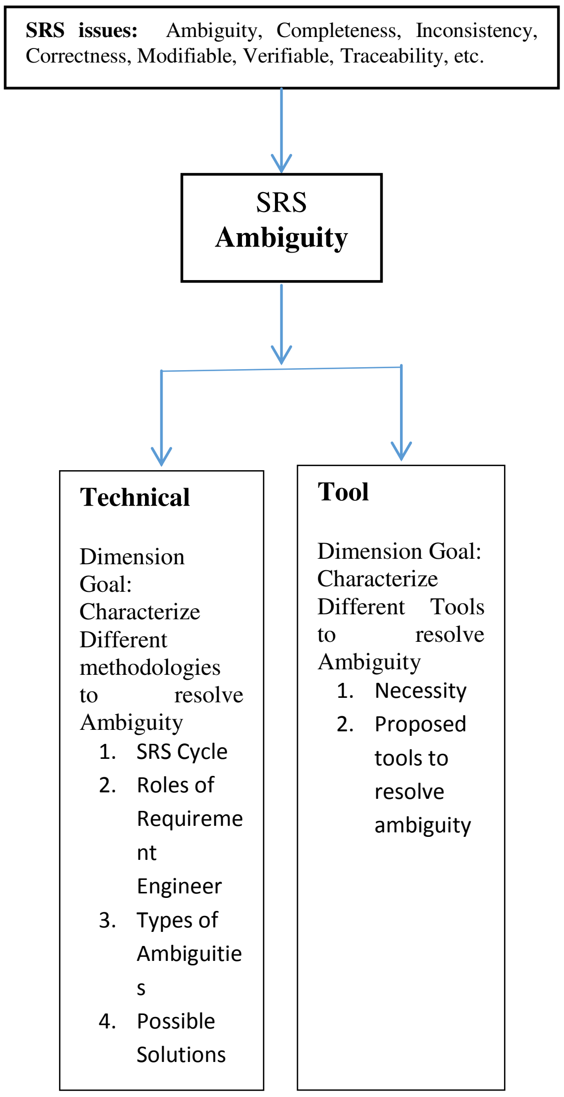
SRS issues: Ambiguity, Completeness, Inconsistency, Correctness, Modifiable, Verifiable, Traceability, etc.
SRS Ambiguity
Technical
Dimension Goal: Characterize Different methodologies to resolve Ambiguity
1. SRS Cycle
2. Roles of Requirement Engineer
3. Types of Ambiguities
4. Possible Solutions
Tool
Dimension Goal: Characterize Different Tools to resolve Ambiguity
1. Necessity
2. Proposed tools to resolve ambiguity

Figure-2. Shows Dimensions and sub dimensions related to SRS ambiguity.

Here we are explaining briefly both dimensions and its related major objectives. The major aim of the technical dimension is to describe the types of ambiguities, methods, techniques, and models to resolve the ambiguity. To achieve this, each approach needs to be described with proper detail as the process of elicitation, the type of ambiguities, the different team roles (team roles, size of the project, and selection), and the technique applied to detect and remove ambiguity in SRS documents. Finally, the tool dimension describes how tools can support to detect and remove SRS issues. For this dimension, we described the role of the various tools (purpose) and investigated what technique is used by the different tools as there are plenty of tools that work on different types of ambiguities. The above two dimensions may not be completely distinct as some of the parameters may be overlapped, as for now, this is unavoidable. For example, techniques discussed in the technology might be used as a base for the tool in the other dimension. Here tools are only meant for the automatic detection and removal of the ambiguities so that we can reduce the overall cost and can save precious time of the team. But still, we tried our best to better clarify these dimensions from the end user perspective. Now we will pursue our discussion on the two dimensions and their sub dimension's with the help of appropriate material which we extracted from the databases by using related keywords.

4. THE TECHNICAL DIMENSION OF SRS AMBIGUITY

It starts with the clarity of SRS life cycle to the requirements engineer or business analyst so they can detect and remove SRS ambiguities by inculcating extra efforts to analysis and validation phase. Final drafted SRS should be complete, correct and unambiguous, for a particular development situation. Detailed description of the technical dimension with existing issues related to SRS ambiguity and refinements, to improve the quality of SRS, is important. As shown in Figure-3, the detail classification of our technical dimension which includes the Requirements Engineering Process, role of various team members and participants, various types of ambiguities and the methods, technique, models to prevent, detect and remove ambiguity as a sub-dimension. More details of each sub-dimensions are discussed here in the below section. We found, in total 58 references related to the technical dimension.

5. REQUIREMENT LIFE CYCLE

Requirement life-cycle involves a number of phases and at times it can be a complex process. The nature of the process depends on the model you choose for your Software development like V, Double V, Waterfall, Incremental, etc.

We surveyed the literature and found various issues in choosing the requirement model. There are factors which can greatly affect the SRS.

a) Project (Size, Complexity, Requirement Uncertainty, Time constraint, Generic, Repetitive or New etc.)
b) Requirement Engineer (Experience with Elicitation, Formal Training, Domain Experience, Training on the same technique etc.)
c) Stakeholders (Experience, Location, Expressiveness, Time Availability etc.)

We are not able to find any rule which can guide the team to choose suitable model, as it all depends on the nature of project and experience of the various stakeholders. In the process of SRS, the major responsibility is on the requirements engineer to extract the requirements from the stakeholders and specify them clearly. There are six well known major Requirements engineering phases that should be known to Business Analyst or Requirements Engineer: planning, requirements gathering, requirements analysis, requirements documentation, requirements validation, requirements management. Requirements development and requirements management are two main activities of requirements engineering. Activities like extraction, analysis, and validation are part of requirements development [17]. Whereas requirements management comprises all activities involved in making changes in requirements baseline request, performing analysis for a change request.

5.1 Requirements planning

In the planning phase, various significant tasks are accomplished by the analyst, i.e. goal identification, prioritization, team organization, assigning of responsibilities and preparation of various materials to be used in different phases of RE process. Responsibility is assigned to various stakeholders for elicitation, analysis, verification and documentation phases [18]. Some materials are prepared that can be used in various phases of the RE process, which includes a list of questions used during the elicitation phase of the requirements engineering.

5.2 Requirement elicitation

Requirement gathering or elicitation is one of the most crucial phases of requirement engineering process commonly known as requirement mining. After gathering the requirement, this product can be arranged logically in folders according to the release [19]. Impact assessment process involves requirements investigation to formulate facts and figures to track possible outcomes based on the analysis.

5.3 Requirements analysis

Discussion and analysis is a process in which requirements are studied in relation to compatibility, possibility, and completeness.

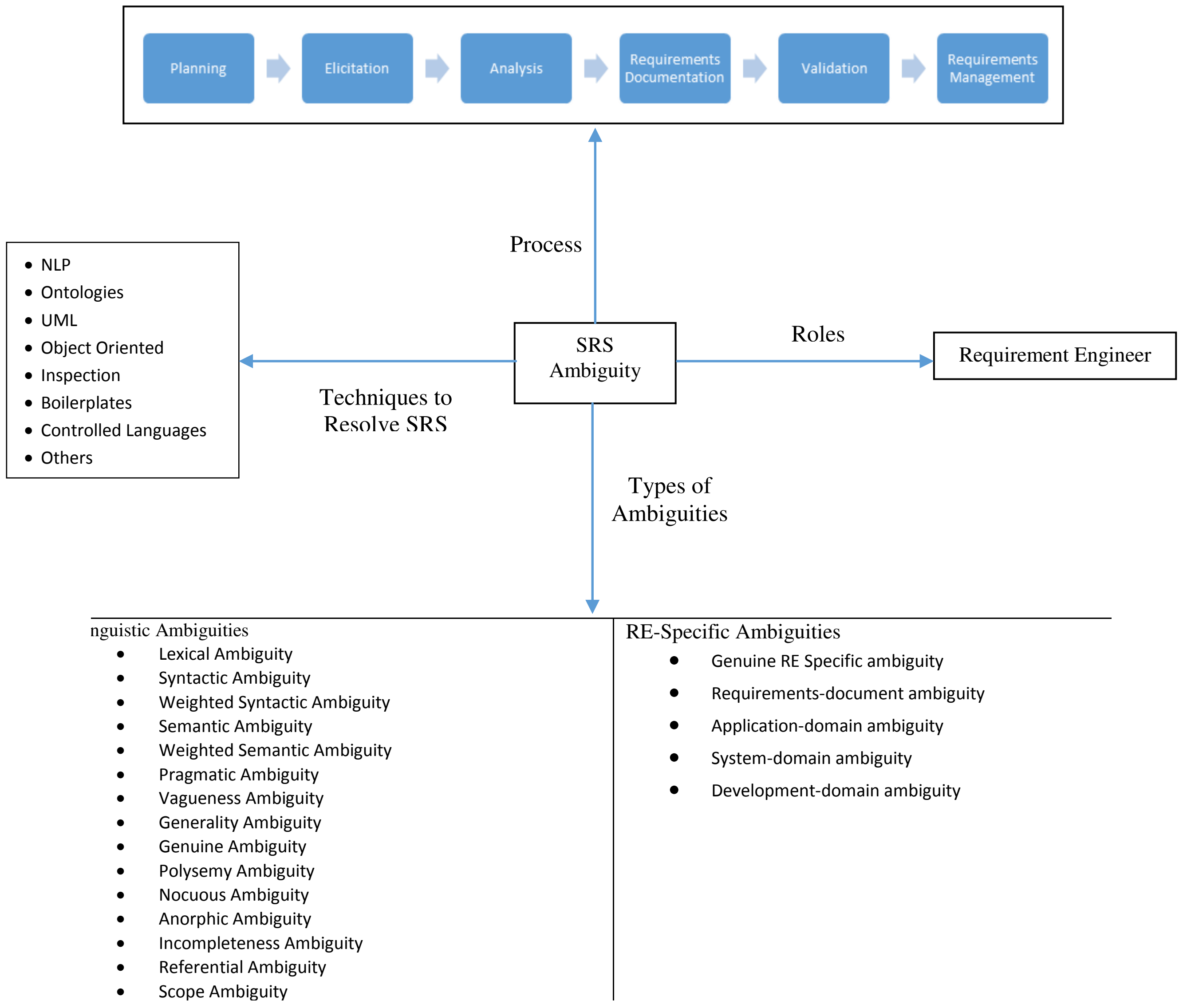

Figure-3. Shows Technical Dimensions of SRS ambiguity.

Prioritizing the discussions to omit requirement interference and identifying the risk of the issues leads to relevant set of output requirements. Techniques for analyzing the requirements are model by Joint applied development (JAD) meetings, modelling, priority requirements and quality function deployment (QFD), using three different types of languages namely official, semi-official and informal language.

As the requirements are met and modelled, problems may arise in maintaining the pacts with all stakeholders, particularly if stakeholders have different goals [20]. The authors believe that collective requirements must go through an analysis process in order to achieve the best requirements [20-22]. Negotiations to resolve conflicts between stakeholders are necessary attempts, to weaken the essential weaknesses of each stakeholder's goal [20]. The objective of this phase is to find the problems that occur in the initial requirements statements of elicitation phase. The following types of investigations are performed in the requirement analysis phase using the analysis checklist (As per authors).

a) Ambiguity Checking
b) Necessity or requirement checking.
c) Completeness checking
d) General comments

Usually, stakeholders analyze the requirements for completeness; means that those requirements which are not needed are omitted [17]. Requirement statements are said to be complete if all the parts are present and there is no existence of "to be defined" statements or postponed decision. Stakeholders examine the requirements to observe whether the requirements achieved are fit for attaining the business goals of the organization. The elicited requirements satisfy the stakeholder's application domain, goals, and objectives of the organizations. It is also learnt whether the elicit requirements achieved are really necessary to solve specific problems. The stakeholders provide general comments at the bottom of analysis checklist, regarding the initial requirements statement. Stakeholders gave suggestions regarding elicited requirements and mention if they agree with the same or if they want further modification then that modification is specified.

5.4 Requirements documentation

The ultimate goal of requirements engineering is to document requirements as described by the customer so that other stakeholders like developers, designers, requirements engineers can understand the exact context of the requirements. Documents of good requirements should be accurate, correct, unambiguous, improvable, revisable and understandable.

Requirements can be considered as the basis for controlling document changes and evaluating the products and processes of future system design, system testing cases and verification. Accepted requirements documents have suitable details as well as suitable symbols. Requirements documentation has a direct relationship with requirements management. The specifications of the requirements can be outlined in small projects in a SRS document or in bigger projects.

- In requirements specification, Functional and non-functional requirements and their limitations are documented.
- **B**usiness requirements document have Business requirements only.
- External interfaces are kept separately in specifications of software requirements or documents of external requirements.

5.5 Requirement validation

Requirement validation is a necessary phase of the requirements engineering and plays a crucial role in the successful execution of any project. Validation of requirements means that it is necessary to ensure that the needs of the customers are complete, and well-written. Requirements validation phase executes the project on the basis of various needs of the stakeholders. Validation in requirements engineering is to control the quality by checking the specification, Traceability Analysis, High Fidelity Simulation, wireframe, etc.

There are numerous requirements validation tools available that can be used for validation with minimal human intervention. If some requirements are incomplete then the same process can be followed through the phases namely, elicitation, analysis, verification and validation for those specific incomplete requirements. If requirements are ambiguous then meetings are held among stakeholders to remove the ambiguities. Tools are also available that can be used to remove the ambiguities. This phase continues to repeat the other stages of development requirements due to lack of identity, the gap between needs, additional information and other issues. The software implemented is valid in the software life cycle test phase based on its requirements [23].

5.6 Requirement management

Requirement Management process includes planning, monitoring, analyzing, communicating and managing requirements. If the requirements are not well managed, then the final product will be adversely affected. Managing thousands of requirements manually seems to be difficult; it is advised to use some tools. A variety of requirement management tool are available online which helps in managing the requirements easily just by putting very small efforts and time.

6. ROLES

There are two important questions practitioners usually have about SRS:

a) What roles are involved in the requirements lifecycle?

b) What skills are required to get complete, correct and unambiguous requirements?

In the first question, specific roles assigned to a participant, therefore each participant has a clear and distinct responsibility as described in the roles and responsibilities [2, 24]. There is not much disagreement between the engineer and Business analyst regarding the role and requirements definitions. The requirements engineering emphasize is on project role, not on the job title. In the following, we will describe the role and responsibilities of requirements engineer in more detail:

6.1 Requirements engineer

The requirements engineer is individual, whose primary obligation is to elicit, analyze, document and validate the complete set of requirements of a system [20]. Requirements engineer/analyst is also known as a requirements manager, business analyst, system analyzer or simply an analyst. The requirements analyst is a bridge between the customer community and the software development team through which requirements flow. Requirements analyst regularly communicates with customer community and development team, also discusses the difficulties with both the parties. Throughout the software development life cycle, the requirements analyst is included at various levels. After achieving the baseline requirements, the focus is transferred towards the management of the requirements specification. Requirements Analyzer is responsible for seeing that the tasks are performed properly and verify the fulfilment of all requirements. In [25] some of the desirable skills are mentioned

a) Interviewing skills, to talk with individuals and groups about their needs and ask the right questions to surface essential requirements information.
b) Listening skills, to understand what people say and to detect what they might be hesitant to say.
c) Analytical skills, to properly investigate and evaluate the information gathered from different sources; decompose high-level information into comprehensive information, to bring a more general understanding of the fundamentals with lower level information.
d) Facilitation skills, to lead requirements elicitation workshops.
e) Observational skills, to validate data obtained via other techniques and expose new areas for elicitation.
f) Writing skills, to communicate information effectively to customers, managers, marketing, and technical staff.
g) Interpersonal skills, to help negotiate priorities and to resolve conflicts among project stakeholders (such as customers, product management, and engineering).
h) Modeling skills, to represent requirements information in graphical forms that enhance textual representations.

Organizational skills, to deal with the huge array information gathered during organizational skills, elicitation and analysis and dealing with rapidly changing information.

7. TYPES OF AMBIGUITIES

According to Kamsties & Peach [26] ambiguities can be resolved if we know the RE context. Further, author took the RE context into consideration and defined an ambiguous requirement is a requirement that has different meanings. He described the importance of RE context because almost all natural language requirements have a high probability to have ambiguous meaning. While reading the requirements most of the requirements can be disambiguated by a reader who understands the context of RE, rest of the requirements, we consider ambiguous. A requirement allows multiple interpretations if it contains linguistic or RE-specific ambiguity (as shown in Table-2). Linguistic ambiguity (Syntactic, Lexical, Semantic, generality and vagueness ambiguity etc.) is independent of any context. RE-specific ambiguity depends on system-domain, application-domain, development-domain and RE context ambiguity [26]. Table-1 shows types of ambiguities with references.

8. TECHNIQUES TO RESOLVE AMBIGUITIES

Here our main focus is to identify existing solutions which can improve the quality of the SRS in terms of ambiguity detection and reduction. We reviewed and analysed 54 literatures and found most of the solutions can be divided into six main categories based on the technique used by researchers to resolve ambiguity.

In the survey, we found that solutions to SRS ambiguity can be broadly categorized into six categories:

a) Ontology Based Solutions
b) Object-Oriented & UML Based Solutions
c) NLP Based Solutions
d) Inspections Based Solutions
e) Boilerplates Based Solutions
f) Controlled Languages Based Solutions

In [27] the study is based on the automatic and semi-automatic tools only, whereas our survey includes all techniques, models, algorithms, and frameworks investigated by the different authors to detect and reduce ambiguity over the period of time. To know about various techniques, we thoroughly reviewed the literature and found that ontology based solutions are commonly used to detect and remove ambiguity. Several types of automated tools are listed to reduce ambiguity are majorly based on ontologies and NLP [10].

The First type of (24%- [24, 28-38]) solution is based on ontologies. Ontology is a collective formal conceptualization of a domain that allows the definition of semantic relationships between entities and inference of knowledge through reasoning [24]. Domain ontologies are suitable for building requirements elicitation and can help reduce the effort. Background knowledge can be used to disambiguation in the unstructured text [35]. In order to recognize a pattern in the problem description, a theoretical model for knowledge of the domain is required so that terms, entities, and relationships among them are clear. Polpinij, Jantima [31] used a text classification and filtering technique to discard irrelevant sentences which have similar details so that we can avoid ambiguity. Further ontology based question answers are helpful to resolve functional and lexical ambiguities [30, 33]. As far as non-functional requirements are concerned Rashwan *et al*. [38] proposed a new corpus & support vector machine-based classifier to examine the sentence based classification of requirements.

The second category is (18% [44-52]) Model-based solution to ambiguity. Models can be based on UML or Object-oriented. Meziane, Athanasakis, & Ananiadou in [44] developed a java based tool "GenNLangUML" to generate Natural Language Specifications using UML diagrams to disambiguate the SRS. In [45] proposed a semi-automatic tool to identify ambiguities in SRS. Natural Language (NL) SRS is parsed by using constrained grammar, then relationships extracted in the parsing, the tool creates methods, classes, and associations and finally, the model is diagrammed so that human reviewer can detect ambiguities. In [52] author proposed and mentioned a method which translates natural language requirements to state transition diagrams in an incremental manner and allows requirements engineers and other stakeholders to participate in the conversion process. This approach can improve requirements during the conversion process by recognizing ambiguities and incompleteness in the NL requirements. LUCSED (Language of Use Case and Sequence Diagram) [51] tool is used to automatically generate use cases and sequence diagram from textual requirements. It mitigates natural language problems like ambiguity. According to Kamalrudin, Hosking, and Gundy in [46] using an essential use case interface pattern library can improve the quality and reduce the inconsistency. Further [47] used a modeling approach known as Restricted Use case modeling which is composed of rules and restrictions and an altered use case model to reduce the ambiguity. The author claimed approach is easy and gives good results over traditional approach in terms of class completeness and sequence diagram completeness. Saurabh and Atul in [49] performed an experiment using UML use case model and formally specified use case models and assessed the later gave better results in terms of correctness and completeness. The formal use case template showed to be better quality analysis model than the semi-formal use case model.

The third category is (16%- [27, 43, 53-58]) based on NLP techniques. In [27] author proposed a NLP technique based on text chunking to automatically verify the templates. Templates can be used to write natural language requirements to avoid ambiguity. When we apply a template, it is vital to authenticate the requirement, which is

Types of Ambiguities	Definition	Articles
Lexical Ambiguity	Lexical ambiguity occurs when a word has several meanings	[10, 12, 30-32]
Syntactic Ambiguity	Syntactic ambiguity, also called structural ambiguity, occurs when a given sequence of words can be given more than one grammatical structure, and each has a different meaning.	[10, 12, 13, 30-32]
Weighted Syntactic Ambiguity	Weighted Syntactic Ambiguity is the number of possible roles weighted according to their frequency.	[13]
Semantic Ambiguity	Semantic ambiguity occurs when a sentence has more than one way of reading it within its context although it contains no lexical or structural ambiguity.	[10, 12, 13, 30-32]
Weighted Semantic Ambiguity	Weighted Semantic Ambiguity is the number of possible roles weighted according to their frequency.	[13]
Pragmatic Ambiguity	Pragmatic ambiguity occurs when a sentence has several meanings in the context in which it is expressed	[10, 31]
Vagueness Ambiguity	Vagueness occurs when a phrase has a single meaning from grammatical point of view, but still leaves room for interpretation, when considered as a requirement.	[10, 30-32]
Generality Ambiguity	Generality, is the other form of indeterminacy e.g. Sue is visiting her cousin. Cousin can be male or female	[10, 30]
Genuine Ambiguity	A requirement is genuinely ambiguous if it has a discrete number of interpretations, no general meaning is available which covers the distinct readings, and clarification is required to make sense of it.	[30]
Polysemy Ambiguity	A word is polysemous when it has several related meanings and one etymology. Polysemies are often of a RE-specific nature. The meanings of a polysemy are related, thus, more detailed contextual information is necessary to disambiguate it.	[30]
Nocuous Ambiguity	Nocuous Ambiguity occurs when a sentence can be interpreted differently by different readers	[33]
Anorphic Ambiguity	Anaphoric ambiguity occurs when the text offers two or more potential antecedent candidates either in the same sentence or in a preceding one.	[34]
Incompleteness Ambiguity	A grammatically correct sentence that provides too little detail to convey specific or needed meaning	[32]
Referential Ambiguity	A grammatically correct sentence with a reference that confuses the reader based on the context	[12, 32]
Scope Ambiguity	When other constituents in its structural context determine the meaning of a constituent in a sentence, we say that the constituent is in the scope of the constituents that determine its reference	[12]

Table 1: Different type of ambiguities listed by authors

RE Specific Ambiguities	Definition	References
Genuine RE Specific ambiguity	A requirement is RE-specifically ambiguous if it has a discrete number of interpretations in relation to the RE context.	[30]
Requirements-document ambiguity	Requirements-document ambiguity occurs because a requirement allows several interpretations with respect to what is known about other requirements in the requirements document.	[30]
Application-domain ambiguity	Application-domain ambiguity occurs because a requirement allows several interpretations with respect to what is known about the application domain.	[30]
System-domain ambiguity	System-domain ambiguity occurs because a requirement allows several interpretations with respect to what is known about the system domain.	[30]
Development-domain ambiguity	Development-domain ambiguity occurs because a requirement allows several interpretations with respect to what is known about the development domain.	[30]

Table 2: RE specific ambiguities

difficult, NLP techniques can be used for the automatic checking of requirements. In [54, 55, 57] natural language pattern is used to increase the quality and to reduce the ambiguity. Domain-dependent patterns are helpful to improve the precision of the requirements specification [54]. While in [57] the definition of linguistic patterns was introduced through a new language known as RSL-PL language, which empowers the linguistic patterns to abstract information to perform the linguistic analysis. Guiding rules and specific language patterns are used to write the Natural Language Requirements, validation of ambiguous requirement sentences can be done using the rules and patterns [55]. These linguistic rules and language pattern become the base of a requirements engineering tool SREE (Systemized Requirements Engineering Environment). A machine learning algorithm is used to find sentence level ambiguity and a prototype tool NAI can be used to reduce the coordination ambiguity and misunderstanding among different team members [58].

The fourth category of (16% [59-66]) solutions are inspections focused. Reviews are conducted during development to refine the SRS to avoid ambiguity. Inspection is used when SRS is supposed to be completed just to detect ambiguities. Effective inspection method can reduce the ambiguity and can improve the quality of Software requirements document. Reading is one of the inspection techniques which can be applied by the reviewer to put more focus on the significant parts of an entity while inspection. Kamsties et al. in [60] proposed a checklist and scenario-based inspection technique. Another type of reading is Usage-based reading focuses on user's point of view of a software artifact to help reviewer so that they can focus only the important parts [61]. As per Berling & Runeson, (2003) [62], perspective based inspection is more effective than checklist based inspection. Tjong, Hartley, & Daniel, (2008) [59] in their thesis gave some guiding rules on the basis of the corpus which can serve as an Inspection checklist to find ambiguities. In [65] author perspective-based inspection methodology is developed based on Pragmatic Quality Model to recognize 198 total inspection points to prepare a quality inspection report.

The fifth category is (14% [14, 18, 24, 67-70]) boilerplate based solutions. A Boilerplate is a textual template for requirements specification, which is based on predefined patterns in order to reduce ambiguity [24]. For requirements, the grammatical structure of sentences can be improved through Boilerplates suggested patterns. A variety of boilerplates have been suggested by the researchers, two such well-known boilerplates are the EARS (Easy Approach to Requirements Specification) boilerplate [68] and Rupp's boilerplate [18]. To decrease the problems in NL specification EARS boilerplate is one of the most commonly used boilerplate. Farfeleder, Stefan *et al.* [67] uses boilerplates in their tool that can semi-automatically transforms the natural language requirements, into semi-formal boilerplates. In [69] author presented a tool-supported approach for boilerplate to check conformance because the checking of conformance manually to boilerplates is very difficult. Apart from this, boilerplate is one of competent technique to decrease ambiguity and become more adaptable for automatic analysis. In addition, it offers a modest and significant way for improving requirements quality in terms of inconsistency, ambiguity and by avoiding complex structure, with the help of Text chunking which provides a precise and correct basis for checking conformance to boilerplates [70].

The sixth type (12% [16, 71-75]) solutions is based on controlled language. Using controlled language means we are trying to avoid the ambiguities. Constrained/Controlled Natural Language (CNL) is becoming popular for writing requirements specification. CNL reduces ambiguity within natural language while maintaining their readability & expressiveness. Combination of constraint lexicons and word sense disambiguation reduced the ambiguity [71]. Controlled language, verification and automatic extraction of requirements models are used to reduce the ambiguity [72]. There are many tools which are based on controlled language like Word Sense Disambiguation (WSD), wiki-based prototype, Attempto Controlled English (ACE) and NL2OCL.

Other types of solutions are:

The reason to keep these solutions into other categories is either the number of solutions is less than two or these are not fit into any of the categories. Stories and Scenario based requirements can ensure the proper, unambiguous and verifiable requirements [76]. Agile requirements gathering is another effective method to understand the words clearly which lead to the less ambiguous SRS [77]. In [78] author proposed a string rewriting technique and overall string amendment rules that can be practiced in the development to automatically make new equalities thereby motivating the human for a smaller amount of decisions which leads to a

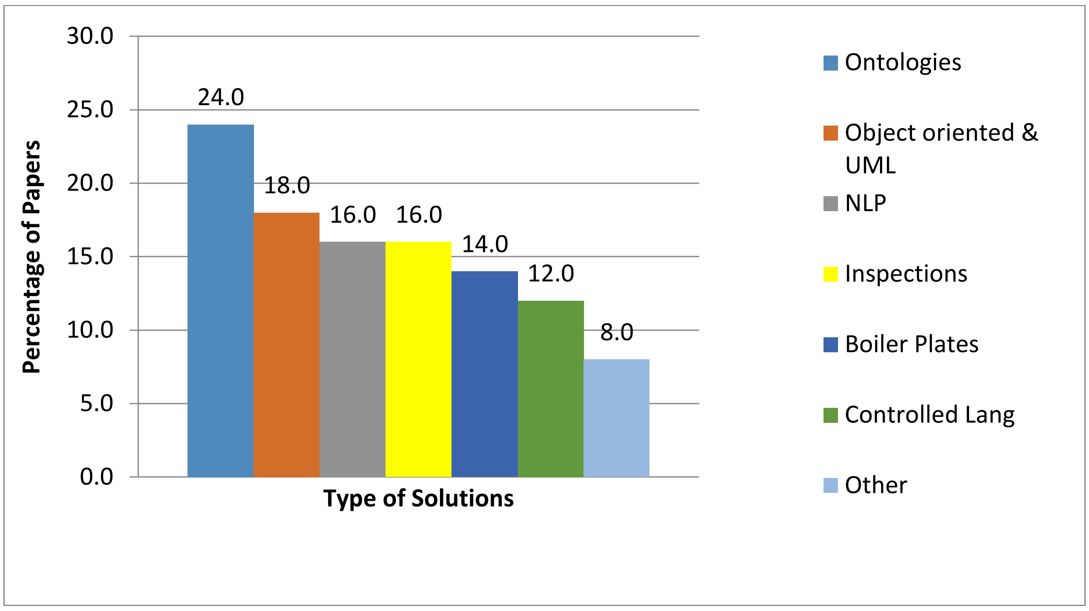

Figure-4. The amount of research to resolve to ambiguity from 2000-2016.

reduction in ambiguity. Figure-4 illustrates a number of papers (in percentage) in the form of a bar chart, which represents the mainly six types of techniques to reduce ambiguity.

9. THE TOOL DIMENSION OF SRS AMBIGUITY

Manually resolving ambiguity from software requirements is a difficult task, time taking process, and that's why it's quite expensive process [39]. Moreover, as systems turn out to handle high-quality commercial applications, their requirements become huge, further growing scope can lead to ambiguity escalation [79]. Therefore, we need some mechanism to resolve ambiguities from the requirements documents. There exist varied approaches, beginning from inspections, controlled natural language, lexicon-based approach to ontology-based approach which automatically reduce the ambiguity from the SRS. In addition, there are a number of different tools as shown in Table-3. Systemized Requirements Engineering Environment (SREE) [80], Word sense disambiguation (WSD) [81], Nocuous Ambiguity Identification (NAI) [58], Quality analyzer for requirements specification (QuaARS) [82], ARM [42], Requirements Engineering Specification Improver (RESI) [83], and Natural Language to Object Constrained Language (NL2OCL) [75] developed to detect and resolve ambiguities.

Parsing a text (using Stanford [84]) is helpful to detect or disambiguate ambiguous words by looking at the syntax - however such techniques are useful for lexical and syntactic ambiguities.

10. ANALYSIS AND DISCUSSIONS

In our survey we found number of papers focused on detecting and removing ambiguities. There is no such technique which can avoid the ambiguities. If we can avoid the ambiguities that can make the overall process of requirements gathering comparatively easy or less complex rather than detecting and removing various ambiguities. There are open questions about what will happen if the requirements are complex or the size of the project is large. We went through different types of tools, algorithms, and models that deal with different type of ambiguities. To measure the performance of technique researchers, calculate precision, recall, and F-Measure, we are not able to find any tool, technique achieving 100% accuracy. There is no such tool which can handle all type of ambiguities. As we know the requirements engineer is responsible for requirements gathering and to write the bug-free SRS. How to improve the skills of a requirement engineer to deal with ambiguities? A survey can be performed to compare industry practices to check the effectiveness of different techniques and how they manage ambiguities? A survey/experiment can be performed to determine the efficiency of different tools. A case study can

VOL. 14, NO. 17, SEPTEMBER 2019 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2019 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

Table-3. List of different tools to handle different types of ambiguities.

Tools	Technique Used	Ambiguity Addressed	Reference	Remarks
WSD (Word Sense Disambiguation) (2000)	NLP and Ontology	Lexical, Semantic, Pragmatic	[81]	Tool is amalgamations of different approaches, and the return of knowledge-based systems via graph-based methods.
QuaARS (Quality analyzer for requirement specification) (2001)	NLP	Lexical, Syntactic, Quality	[82]	Automatic tool to evaluate Natural language written requirements
Object Oriented Visualization (2006)	OOP	Lexical, Pronoun	[29]	Useful to extract Non-functional requirements.
SREE (Systemized Requirements Engineering Environment) (2008)	NLP, Rule Based	Identifying Plurals, Pronoun	[59, 85]	Used for ambiguous and incomplete requirements
RESI (Requirements Engineering Specification Improver) (2009)	Ontology	Avoid Lexical Scope	[83]	Automatic UML improver, Input must be in GrGen Graph
NAI (Nocuous Ambiguity Identification) (2010)	NLP	Nocuous, Anaphoric and coordination Ambiguity	[42, 58]	Effective in the case of coordination Ambiguity
SBVR Tools (Semantics of Business Vocabulary and Rules) (2011)	Controlled Language	Lexical, Syntactic	[16]	SBVR rule generation
NL2OCL (Natural Language to Object Constrained Language) (2012)	Knowledge Based and ontology	Homonymy, Syntactic	[75]	A UML class input is required
CKCO (Context Knowledge & Concepts Ontology) (2012)	Knowledge Based and ontology	Lexical - Polysemy (ambiguity	[63]	Resolving ambiguity based on Question Answers based on Context

be launched to compare different tools and techniques on the common data. More appropriately the complications of the ambiguity need to be register and rectified using different techniques, on common SRS, so that comparisons can be made to determine a best possible solution. Most important what is the acceptable ambiguity intensities for requirement documents written in natural language? Need of the day is generic tools/technique to eliminate ambiguities or a model to avoid ambiguity.

11. CONCLUSIONS

The software project success fundamentally depends upon the quality of SRS document. In order to ensure the quality of SRS, we need to examine various quality attributes such as ambiguity, accuracy, and completeness. Here we made survey on one of the quality attributes i.e. ambiguity because ambiguity is the most researched problem. The aim of the survey was consolidation of the large work in the area of SRS Ambiguity.

The survey presented a systematic explanation of the basic concepts and their relationships, which define the different possible components in the field of SRS ambiguity. There are several benefits of this type of survey for experts and researchers. Firstly, it helps to provide organized information, in the form of a vocabulary that helps in the identification of existing SRS ambiguity detection models, methods, techniques, and tools. It also supports to recognize the elements of the suitable approach for a specific situation via learning various dimensions. The survey offers the idea of the ambiguity detection and correction work so far completed and helps experts and researchers describe the type of new work in the SRS ambiguity field, which can be pursued. This investigation also helps to outline a common terminology that describes the ambiguity in SRS. Secondly, it provides the summary of the existing state of research performed in the area and knowledge analysis in the field of SRS ambiguity.

We know that each survey has its own limitations, as it can provide only the snapshot of the current work in progress. Moreover, the survey represents a snapshot or a small portion of the existing articles that are available on the SRS ambiguity. Although, in our case, we examined close to two hundred research articles. Undoubtedly the survey represents a good portrait of the SRS ambiguity related work. In this paper, we tried to categorize the different solutions to resolve ambiguity into mainly six categories depending on the technique used in the solutions. Secondly, we tried to find the number of solutions in each category. We found that ontology is most frequently used to resolve ambiguity, the reason is, its applicability to clearly establish non-functional requirements, which may be difficult otherwise.

Additional research can be performed specifically for the quality comparison of the different techniques. Moreover, we plan to introduce a system which can avoid ambiguity rather than detecting and rectifying it.

REFRENCES

[1] J. Scheffczyk, U. M. Borghoff, A. Birk, and J. Siedersleben. 2005. Pragmatic consistency management in industrial requirements specifications. in Software Engineering and Formal Methods, 2005. SEFM 2005. Third IEEE International Conference on. pp. 272-281.

[2] A. Bucchiarone, S. Gnesi and P. Pierini. 2005. Quality analysis of NL requirements: an industrial case study. in 13th IEEE International Conference on Requirements Engineering (RE'05). pp. 390-394.

[3] F. Belfo. 2012. People, organizational and technological dimensions of software requirements specification. Procedia Technology. 5: 310-318.

[4] P. Heck and P. Parviainen. 2008. Experiences on analysis of requirements quality. in Software Engineering Advances, 2008. ICSEA'08. The Third International Conference on. pp. 367-372.

[5] F. J. Shull and V. R. Basili. 1998. Developing techniques for using software documents: a series of empirical studies. Research directed by Dept. of Computer Science. University of Maryland, College Park, Md.

[6] I. C. S. S. E. S. Committee and I.-S. S. Board. 1998. IEEE Recommended Practice for Software Requirements Specifications.

[7] D. C. Gause and G. M. Weinberg. 1989. Exploring requirements: quality before design: Dorset House Pub. New York.

[8] M. H. Grenat and M. M. Taher. 2008. On the translation of structural ambiguity. Al-Satil J, pp. 9-19.

[9] L. Mich. 2001. Ambiguity identification and resolution in software development: a linguistic approach to improving the quality of systems. in Seventh Workshop on Empirical Studies of Software Maintenance. p. 7.

[10] U. S. Shah and D. C. Jinwala. 2015. Resolving ambiguities in natural language software requirements: a comprehensive survey. ACM SIGSOFT Software Engineering Notes. 40: 1-7.

[11] K. Petersen, R. Feldt, S. Mujtaba and M. Mattsson. 2008. Systematic mapping studies in software engineering. in 12th international conference on evaluation and assessment in software engineering.

[12] [12] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner and M. Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. Journal of systems and software. 80: 571-583.

[13] V. Pekar, M. Felderer and R. Breu. 2014. Improvement Methods for Software Requirement Specifications: A Mapping Study. in Quality of Information and Communications Technology (QUATIC), 2014 9th International Conference on the. pp. 242-245.

[14] U. Anuar, S. Ahmad and N. A. Emran. 2015. A simplified systematic literature review: Improving Software Requirements Specification quality with boilerplates. in 2015 9th Malaysian Software Engineering Conference (MySEC). pp. 99-105.

[15] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry. 2008. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements engineering. 13: 207-239.

[16] A. Umber and I. S. Bajwa. 2011. Minimizing ambiguity in natural language software requirements specification. in Digital Information Management (ICDIM), 2011 Sixth International Conference on. pp. 102-107.

[17] G. Kotonya and I. Sommerville. 1998. Requirements engineering: processes and techniques: Wiley Publishing.

[18] K. Pohl. 2010. Requirements engineering: fundamentals, principles, and techniques: Springer Publishing Company, Incorporated.

[19] E. Kabaale and G. M. Kituyi. 2015. A theoretical framework for requirements engineering and process improvement in small and medium software companies. Business Process Management Journal. 21: 80-99.

[20] B. Nuseibeh and S. Easterbrook. 2000. Requirements engineering: a roadmap. in Proceedings of the Conference on the Future of Software Engineering. pp. 35-46.

[21] B. H. Cheng and J. M. Atlee. 2007. Research directions in requirements engineering. in 2007 Future of Software Engineering. pp. 285-303.

[22] L. Jiang, A. Eberlein and B. H. Far. 2005. Combining requirements engineering techniques-Theory and case study. in Engineering of Computer-Based Systems, 2005. ECBS'05. 12th IEEE International Conference and Workshops on the. pp. 105-112.

[23] M. Attarha and N. Modiri. 2011. Focusing on the importance and the role of requirement engineering. in Interaction Sciences (ICIS), 2011 4th International Conference on. pp. 181-184.

[24] O. Daramola, G. Sindre and T. Stalhane. 2012. Pattern-based security requirements specification using ontologies and boilerplates. in Requirements Patterns (RePa), 2012 IEEE Second International Workshop on. pp. 54-59.

[25] L. Macaulay. 1996. Requirements for requirements engineering techniques. in Requirements Engineering, 1996., Proceedings of the Second International Conference on. pp. 157-164.

[26] E. Kamsties and B. Peach. 2000. Taming ambiguity in natural language requirements. in Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications.

[27] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. 2015. Automated checking of conformance to requirements templates using natural language processing. IEEE transactions on Software Engineering. 41: 944-968.

[28] H. Hu, L. Zhang and C. Ye. 2010. Semantic-based requirements analysis and verification. in Electronics and Information Engineering (ICEIE), 2010 International Conference On. pp. V1-241-V1-246.

[29] G. A. Mala and G. Uma. 2006. Object oriented visualization of natural language requirement specification and NFR preference elicitation. IJCSNS. 6: 91.

[30] A. Rashwan. 2012. Semantic analysis of functional and non-functional requirements in software requirements specifications. in Canadian Conference on Artificial Intelligence. pp. 388-391.

[31] J. Polpinij. 2009. An ontology-based text processing approach for simplifying ambiguity of requirement specifications. in Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific. pp. 219-226.

[32] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, I. Omoronyia and H. Zojer. 2011. Ontology-driven guidance for requirements elicitation. in Extended Semantic Web Conference. pp. 212-226.

[33] C. Unger and P. Cimiano. 2011. Representing and resolving ambiguities in ontology-based question answering. in Proceedings of the TextInfer 2011 Workshop on Textual Entailment. pp. 40-49.

[34] I. Omoronyia, G. Sindre, T. Stålhane, S. Biffl, T. Moser and W. Sunindyo. 2010. A domain ontology building process for guiding requirements elicitation. in International working conference on requirements engineering: Foundation for software quality. pp. 188-202.

[35] J. Hassell, B. Aleman-Meza and I. B. Arpinar. 2006. Ontology-driven automatic entity disambiguation in unstructured text. in International Semantic Web Conference. pp. 44-57.

[36] J. Gracia, V. Lopez, M. d'Aquin, M. Sabou, E. Motta and E. Mena. 2007. Solving semantic ambiguity to improve semantic web based ontology matching," in Proceedings of the 2nd International Conference on Ontology Matching. 304: 1-12.

[37] M. Bhatia, A. Kumar and R. Beniwal. 2016. Ontology based framework for detecting ambiguities in software requirements specification. in Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on. pp. 3572-3575.

[38] A. Rashwan, O. Ormandjieva and R. Witte. 2013. Ontology-based classification of non-functional requirements in software specifications: a new corpus and svm-based classifier. in Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual. pp. 381-386.

[39] A. Handbook. 2003. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity.

[40] B. Gleich, O. Creighton and L. Kof. 2010. Ambiguity detection: Towards a tool explaining ambiguity sources. in International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 218-232.

[41] A. K. Massey, R. L. Rutledge, A. I. Antón, and P. P. Swire. 2014. Identifying and classifying ambiguity for regulatory requirements. in Requirements Engineering Conference (RE), 2014 IEEE 22nd International. pp. 83-92.

[42] A. Willis, F. Chantree and A. De Roeck. 2008. Automatic identification of nocuous ambiguity. Research on Language and Computation. 6: 355-374.

[43] H. Yang, A. De Roeck, V. Gervasi, A. Willis and B. Nuseibeh. 2011. Analysing anaphoric ambiguity in natural language requirements. Requirements engineering. 16: 163.

[44] F. Meziane, N. Athanasakis and S. Ananiadou. 2008. Generating Natural Language specifications from UML class diagrams. Requirements Engineering. 13: 1-18.

[45] D. Popescu, S. Rugaber, N. Medvidovic and D. M. Berry. 2008. Improving the quality of requirements specifications via automatically created object-oriented models. Innovations for requirements engineering. Vol. 71.

[46] M. Kamalrudin, J. Hosking, and J. Grundy. 2011. Improving requirements quality using essential use case interaction patterns. in Proceedings of the 33rd International Conference on Software Engineering. pp. 531-540.

[47] T. Yue, L. C. Briand and Y. Labiche. 2013. Facilitating the transition from use case models to analysis models: Approach and experiments. ACM Transactions on Software Engineering and Methodology (TOSEM). 22: 5.

[48] I. Sayar and M. T. Bhiri. 2014. From an abstract specification in event-b toward an UML/OCL model. in Proceedings of the 2nd FME Workshop on Formal Methods in Software Engineering. pp. 17-23.

[49] S. Tiwari and A. Gupta. 2014. Does increasing formalism in the use case template help? in Proceedings of the 7th India Software Engineering Conference. p. 6.

[50] R. A. Ahmad. 2016. Interface-Driven Software Requirements Analysis. European Scientific Journal, ESJ. Vol. 12.

[51] M. A. Miranda, M. G. Ribeiro, R. D. Tavares, T. H. Dias, H. T. Marques-Neto and M. A. Song. 2016. An Approach for Generating Class and Sequence Models. in Proceedings of the International Conference on Software Engineering Research and Practice (SERP). p. 253.

[52] D. Aceituna, G. Walia, H. Do and S.-W. Lee. 2014. Model-based requirements verification method: Conclusions from two controlled experiments. Information and Software Technology. 56: 321-334.

[53] O. Ormandjieva, I. Hussain, and L. Kosseim. 2007. Toward a text classification system for the quality assessment of software requirements written in natural language. in Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting. pp. 39-45.

[54] C. Denger, D. M. Berry and E. Kamsties. 2003. Higher quality requirements specifications through natural language patterns. in Software: Science, Technology and Engineering, 2003. SwSTE'03. Proceedings. IEEE International Conference on. pp. 80-90.

[55] S. F. Tjong, N. Hallam and M. Hartley. 2006. Improving the quality of natural language requirements specifications through natural language requirements patterns. in Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on. pp. 199-199.

[56] C. Huertas and R. Juárez-Ramírez. 2012. NLARE, a natural language processing tool for automatic requirements evaluation. in Proceedings of the CUBE International Information Technology Conference. pp. 371-378.

[57] D. de Almeida Ferreira and A. R. da Silva. 2013. RSL-PL: A linguistic pattern language for documenting software requirements. in Requirements Patterns (RePa), 2013 IEEE Third International Workshop on. pp. 17-24.

[58] H. Yang, A. Willis, A. De Roeck and B. Nuseibeh. 2010. Automatic detection of nocuous coordination ambiguities in natural language requirements. in Proceedings of the IEEE/ACM international

conference on Automated software engineering. pp. 53-62.

[59] S. F. Tjong, M. Hartley and M. B. Daniel. 2008. Avoiding Ambiguities in Requirements Specifications.

[60] E. Kamsties, D. M. Berry and B. Paech. 2001. Detecting ambiguities in requirements documents using inspections. in Proceedings of the first workshop on inspection in software engineering (WISE'01). pp. 68-80.

[61] T. Thelin, P. Runeson and C. Wohlin. 2003. An experimental comparison of usage-based and checklist-based reading. IEEE Transactions on Software Engineering. 29: 687-704.

[62] T. Berling and P. Runeson. 2003. Evaluation of a perspective based review method applied in an industrial setting. IEE Proceedings-Software. 150: 177-184.

[63] O. Al-Harbi, S. Jusoh and N. Norwawi. 2012. Handling Ambiguity Problems of Natural Language Interface for Question Answering Answering Answering.

[64] F. Salger, G. Engels and A. Hofmann. 2009. Inspection effectiveness for different quality attributes of software requirement specifications: An industrial case study. in Software Quality, 2009. WOSQ'09. ICSE Workshop on. pp. 15-21.

[65] S. Saito, M. Takeuchi, M. Hiraoka, T. Kitani, and M. Aoyama. 2013. Requirements clinic: Third party inspection methodology and practice for improving the quality of software requirements specifications. in 2013 21st IEEE International Requirements Engineering Conference (RE). pp. 290-295.

[66] M. Fagan. 2002. Reviews and inspections. Software Pioneers–Contributions to Software Engineering. pp. 562-573.

[67] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, H. Zojer and C. Panis. 2011. DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development. in Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on. pp. 271-274.

[68] A. Mavin, P. Wilkinson, A. Harwood and M. Novak. 2009. Easy approach to requirements syntax (EARS)," in Requirements Engineering Conference, 2009. RE'09. 17th IEEE International. pp. 317-322.

[69] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer and R. Gnaga. 2013. Automatic checking of conformance to requirement boilerplates via text chunking: An industrial case study. in Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on. pp. 35-44.

[70] C. Arora, M. Sabetzadeh, L. C. Briand and F. Zimmer. 2014. Requirement boilerplates: Transition from manually-enforced to automatically-verifiable natural language patterns. in Requirements Patterns (RePa), 2014 IEEE 4th International Workshop on. pp. 1-8.

[71] S. Boyd, D. Zowghi and V. Gervasi. 2007. Optimal-constraint lexicons for requirements specifications. in International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 203-217.

[72] D. de Almeida Ferreira and A. R. da Silva. 2009. A controlled natural language approach for integrating requirements and model-driven engineering. in Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on. pp. 518-523.

[73] N. E. Fuchs, K. Kaljurand and T. Kuhn. 2008. Attempto controlled english for knowledge representation. in Reasoning Web, ed: Springer. pp. 104-124.

[74] A. Weissman, M. Petrov and S. K. Gupta. 2011. A computational framework for authoring and searching product design specifications. Advanced Engineering Informatics. 25: 516-534.

[75] I. S. Bajwa, M. Lee and B. Bordbar. 2012. Resolving syntactic ambiguities in natural language specification of constraints. in International Conference on Intelligent Text Processing and Computational Linguistics. pp. 178-187.

[76] D. Firesmith. 2004. Generating Complete, Unambiguous, and Verifiable Requirements from Stories, Scenarios, and Use Cases. Journal of Object Technology. 3: 27-40.

[77] D. Gordon, D. Lawless and C. Gordon. 2014. Speak Clearly, If You Speak at All; Carve Every Word

before You Let It Fall: Problems of Ambiguous Terminology in eLearning System Development. Irish Journal of Academic Practice. 3: 9.

[78] R. Eschbach, L. Lin and J. H. Poore. 2013. Applying string-rewriting to sequence-based specification. Formal Methods in System Design. 43: 414-449.

[79] P. Zave. 1997. Classification of research efforts in requirements engineering. ACM Computing Surveys (CSUR). 29: 315-321.

[80] S. F. Tjong. 2008. Avoiding ambiguity in requirements specifications. Citeseer.

[81] N. Ide and J. Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. Computational linguistics. 24: 2-40.

[82] F. Fabbrini, M. Fusani, S. Gnesi and G. Lami. 2001. The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. in Software Engineering Workshop, 2001. Proceedings. 26th Annual NASA Goddard. pp. 97-105.

[83] S. J. Korner and T. Brumm. 2009. Resi-a natural language specification improver. in Semantic Computing, 2009. ICSC'09. IEEE International Conference on. pp. 1-8.

[84] D. M. Cer, M.-C. De Marneffe, D. Jurafsky and C. D. Manning 2010. Parsing to Stanford Dependencies: Trade-offs between Speed and Accuracy. in LREC.

[85] S. F. Tjong and D. M. Berry. 2013. The design of SREE-a prototype potential ambiguity finder for requirements specifications and lessons learned. in International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 80-95.