A Simplified Systematic Literature Review: Improving Software Requirements Specification Quality with Boilerplates

Umairah Anuar[1], Sabrina Ahmad[2]
Innovative Software System and Services (IS3),
Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, 76100, Melaka, Malaysia
umairahanuar88@gmail.com[1],
sabrinaahmad@utem.edu.my[2]

Nurul A. Emran
Computational Intelligence Technologies (CIT) Lab,
Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, 76100, Melaka, Malaysia
nurulakmar@utem.edu.my

***Abstract*— The quality of Software Requirements Specification (SRS) is crucial in order to ensure successful project completion. SRS of poor quality usually lacks of quality attributes such as completeness, accuracy and disambiguity. Boilerplate is a technique used to deal with problems in SRS. However, study on the coverage of boilerplate's contribution especially in improving SRS quality is limited. This paper presents Systematic Literature Review (SLR) on problems in SRS and boilerplates. The review that covers literature from 1997 to 2015 reveals that 1) poor quality SRS is the most popular problem among the other five SRS problems discovered, 2) Boilerplate technique has been applied to cope with SRS of poor quality, where disambiguity has been found the most popular quality attribute.**

Keywords*— *software requirements specification, SRS quality, boilerplates

I. INTRODUCTION

Requirement is a foundation of a system that moulds the shape of the system to be developed in a software project. Software Requirements Specification (SRS) is a fundamental document which consists of a set of requirements that forms the foundation of software development process. SRS does not only list the requirements of a software system but also has a description of its major features. SRS is important as it describes the stakeholders' needs and it provides a basis for developing the software design. In addition, SRS is referred to design test cases for the purpose of verification and validation. SRS usually defines how an application will interact with system hardware, other programs and users. Ideally, most of the developers' queries about an application are answered in the SRS. SRS of high quality clearly describes functional and non-functional requirements which provides sufficient information for the software architect, designer, developer and tester to pursue. Typically, SRS constitutes the agreement between clients and developers regarding the contents of the to-be-developed software product.

However, in reality, many SRS documents are filled with badly written requirements that lack of quality attributes such as completeness, accuracy and disambiguity [4]. Poor quality SRS does not only lead to increase in development and sustainment costs but also cause major schedule overruns [5].

Several techniques have been proposed to improve the quality of SRS (e.g., inspection method, checklist, requirement tool and walkthrough) where boilerplate has been recognised as one of those. Historically, the term boilerplate is used to refer to the sheet steel used to make boilers in the field of printing [54]. Printing plates of text for widespread reproduction such as advertisement or syndicated columns were cast or stamped in steel. In contract law, the term boilerplate describes the parts of a contract that are considered standard [55]. Nevertheless, in computer programming, the term boilerplate is used to represent the section of code (in form of text). Boilerplate is similar to a template except that a template holds layout and style information. The main advantage of boilerplate is reusability where it can be used in several places in a computer programme with little (or no) alteration [56]. While boilerplate promotes formalism, it is also used as a preliminary basis for requirements checking. It also provides a simple and yet effective way for increasing the quality of requirement by avoiding complex structure, ambiguity and inconsistency in requirements [15]. Boilerplates also can be used for several classes of requirements such as capability, function, timeliness, mode and operational constraint [26]. Fig.1 shows an example of boilerplates (called as Rupp's boilerplate) used in requirements collection [57].

978-1-4673-8227-4/15/$31.00 ©2015 IEEE

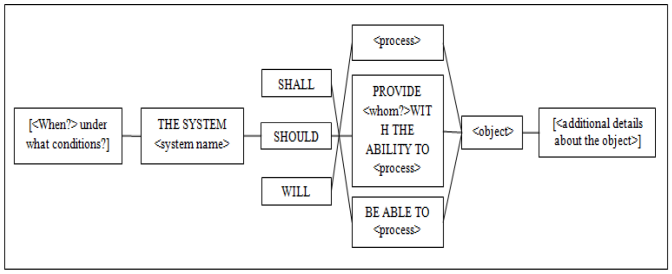

Fig. 1. Rupp's Boilerplate [57].

Boilerplate's contribution in improving the quality of SRS can be seen frequently in the literature. However, studies that examine the coverage of boilerplate's contribution in improving the quality of SRS are limited. Understanding on the quality attributes that have been addressed by boilerplate is important in order to identify gaps in this area. Therefore, this paper aims to present a systematic review of literature on problems in SRS and the contribution of boilerplates in improving the quality of SRS documents.

The rest of this paper is organized as follow. Section II presents SLR method, where the formulation of research questions are described, Section III presents the findings and discussion of the findings. Finally, Section IV concludes the paper.

II. SYSTEMATIC LITERATURE REVIEW METHOD

In this research, review on problems of SRS and boilerplate is conducted using a review technique as proposed by Kitchenham & Charters, called as Systematic Literature Review (SLR) [35]. In software engineering research, SLR is commonly used by researchers to systematically analyze the literatures. SLR creates a connection between an existing knowledge and the problem that needs to be solved. This method includes the process of identifying, accessing and interpreting all available research evidence (mainly literatures) with the purpose to provide answers for specific research questions.

In this section, we present research question formulation, review protocol implementation; data extraction and synthesis.

A. Research Questions Formulation

The formulation of research question is driven by the following motivations:

1) To understand and to discover problems in SRS.
2) To understand and to discover contribution of boilerplates in SRS quality.

The following are research questions that arise from the research motivation stated above:

1) Research Question 1: What are the problems related to SRS?
2) Research Question 2: How boilerplate contributes in improving SRS quality?

B. Review Protocol Implementation

The review protocol specifies methods that will be used to undertake the SLR. Below are protocols applied to conduct SLR in this paper.

1) Source Selection

The purpose of source selection and search is to find relevant sources of literatures within the research area. The sources of the literatures are digital libraries and databases. The digital databases are the most popular and familiar source to find the set of related literatures. As shown in Table I, there are three main sources of SLR used in the research, which are database, Journals and other sources (i.e books, thesis and SRS samples).

TABLE I. SOURCES OF SLR

Database	IEEE Explore, ScienceDirect, Springer, Google Scholar, ACM Digital Library, Scopus, Google Pattern, Cornell University Library, Elsevier
Journals	Empirical Software Engineering – An International Journal Requirements Engineering Journal
Other Sources	Requirement Engineering Books, Science Computing Books, doctoral thesis, SRS samples.

2) Inclusion and exclusion criteria creation

Table II shows the inclusion and exclusion sets in conducting the SLR. The purposes of inclusion and exclusion criteria are to ensure that only relevant papers are included.

TABLE II. SOURCES OF SLR

Inclusion Criteria	Exclusion Criteria
Studies on software requirement specification.	Studies that do not focus on software requirement specification.
Indexed journal papers and indexed conference paper.	Short papers, tutorial and summaries of keynote or workshop.
Indexed journal paper and indexed conferences paper of SRS written in English language.	Indexed journal paper and indexed conference paper written in non-English language.
Studies on boilerplates to improve SRS quality.	Studies that discuss defects and quality approach, frameworks in contexts other than SRS.
Quality attributes for SRS.	Quality attributes for software.
Other sources such as SRS samples, books and thesis.	Other sources that do not contain SRS information.

3) Literature Search and Selection

Literature search and selection process is conducted by following an adapted process flow as shown in Fig.2. We adapt the process flow proposed by Kitchenham and Charters [35] by adding steps in search keyword formulation. In our research, search keyword formulation are performed in two levels. In the first level, the first set of search keywords are formulated based on the research questions; where some of the keywords are augmented with synonyms. The first set of search keyword strings (K_1) formulated is K_1 = { "software

requirements specification", "problem in SRS", "quality in SRS", "boilerplate", "natural language" } . Based on these keywords, we search the literature from the selected sources. Selection of relevant literature are made by checking the Inclusion and Exclusion Criteria as shown in Fig. 2.

Once the relevant literature are retrieved, we conducted the second level of search keyword formulation. The second set of search keyword is formulated based on the contents of the literature retrieved earlier. We also refer some keywords stated in these literature (i.e journals and conference papers) in order to formulate the second set of search keyword. Most of the keywords in the second set consists of terms used to represent quality attributes for SRS. As the result, the second set of search keyword strings (K_2) is K_2 = {"unambiguous", "complete", "correct", "understandable", "verifiable", "consistent", "achievable", "concise", "design independent", "traceable", "modifiable", "interpretable", "stablility", "not redundant", "precise"," reusable" , "organized" }. By using K_2, the second set of literatures are retrieved and the selection process is made by performing Inclusion and Exclusion Criteria checking as in the previous step. These literatures are combined with the literatures retrieved earlier and become the final set of literature that are used to answer the research questions.

C. *Data Extraction and Synthesis*

We extract information from the final set of literatures to answer the research questions. Two groups of data items extracted from the literatures: generic data items as shown in Table III; data items for specific research focus as shown in Table IV. The generic data items are extracted based on K_1, while data items extracted for research focus are based on K_2. More data items of K_2 can be referred in Appendix.

TABLE III. GENERIC DATA ITEMS

Data Items	Description
Bibliography	Author, year, title, source
Type of Article	Journal/ conference paper/ technical report
Study Aim	The aims or goals of the primary study
Methods	The methods used in the paper

TABLE IV. DATA ITEMS EXTRACTED FOR RESEARCH FOCUS

Search Focus	Description	
"SRS"	Focus	Focus on approach/ technique/ method used to produce SRS.
	Approach	Identify on approach used in producing SRS.
"Problem in SRS"	Focus	Problem that exist in SRS.
	Approach	Approach that has been done to reduce the problem exists in SRS.
"Quality of SRS"	Focus	Quality attributes in SRS.
	Approach	Approach that exist to improve quality of SRS.
"Boilerplate"	Focus	The advantages of boilerplates.
	Approach	Approach that has been done to reduce problem is SRS using boilerplates.
"Natural Language (NL)"	Focus	Problem that arise from NL.
	Approach	Approach that has been done to overcome NL problem.

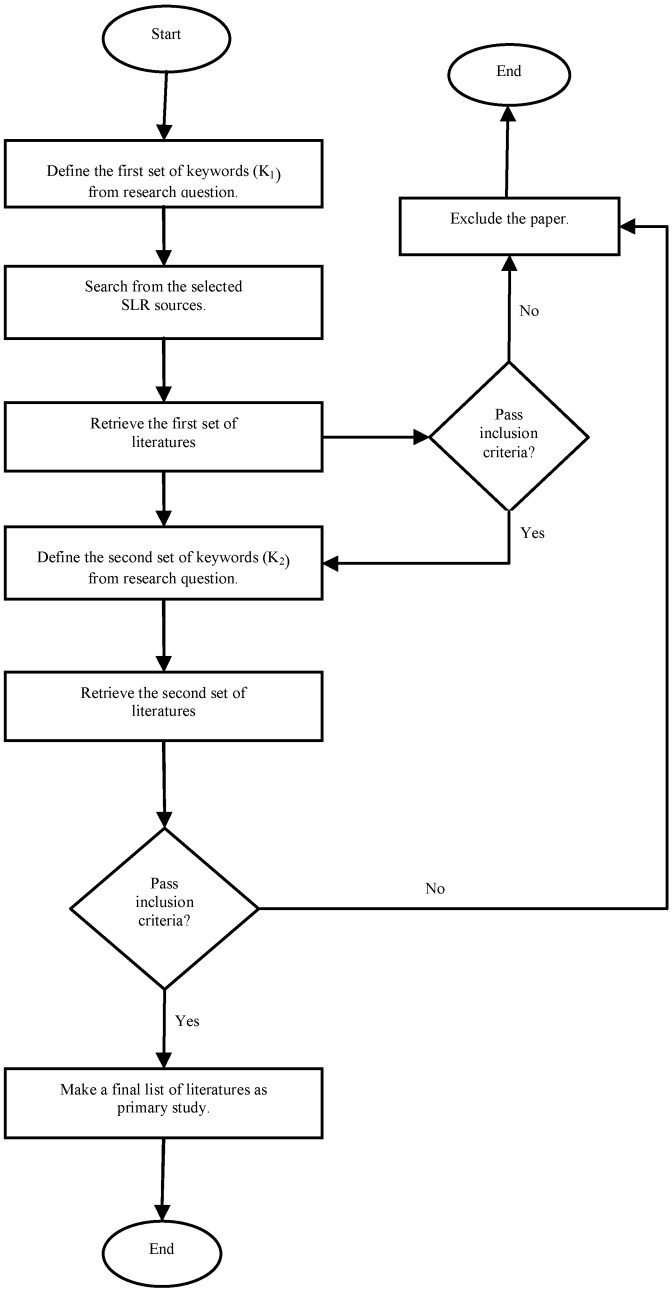

Fig. 2. Literature search and selection process (adapted from [35])

III. FINDINGS AND DISCUSSION

In this section, we present the findings of SLR according to the research questions.

A. *Findings for Research Question 1: "What are the problems related to SRS?"*

In answering this research question, we reviewed 51 literatures that state problems of SRS. We found that, problem

in SRS can be divided into six main categories as shown in Fig. 3. The first problem (which is the most popular) is SRS of poor quality. Several types of quality problems such as ambiguity, incompleteness, incorrectness, irrelevant, lacking in important data, untraced and unverifiable are documented as contributors of poor quality SRS [5]. Most of these problems arise because engineers are inadequately trained and have inadequate access to the sources of the requirements [53]. In addition, traceability is also marked as a quality attribute as the failure to document sources of requirements leads to difficulty in assessing the impacts of changes of requirements [5].

The second problem of SRS is natural language (NL) problem. NL is being used widely to translate the stakeholders' needs into a formal documentation but it poses some disadvantages. NL is prone to ambiguity and without enforcing restrictions, it can be hard to analyze the requirements [20, 23]. Requirements engineering mostly concerned with communicating with customer to gather essential information of the requirement. In order to translate the stakeholders' wants and the system's needs, requirement elicitation and requirement analysis need to be carried out. These activities involved the usage of an informal NL [19].

Insufficient requirements management is the third problem identified in SRS. When developers stored the requirements in paper documents or in simple spreadsheets, it will be hard to find, sort, query and maintain the requirements [5]. Requirements stored in a paper form are difficult to create, manipulate and maintain. There are often missing important data such as priority, type, status, source and rationale. An excessive requirements change is one of the contributors [10]. In addition, unmanaged and unexpected changes can raise havoc with existing architectures, designs, implementations and testing. Without a minimum amount of stability, it is difficult for developers to do their jobs and deliver new systems or increments to existing systems [11].

Insufficient tool support is another SRS problems that usually occur during the maintenance of SRS. Many developers are still using SRS as requirements repository or using simple spreadsheet and relational database table [9], which unfortunately leads to poor SRS maintenance.

The fifth problem is insufficient verification in SRS. Requirements defects that are not identified during the requirements engineering process will give negative impact to the subsequent activities in the software development lifecycle [6]. Therefore, verification should be done early in the development process even if the requirements have sufficient quality to avoid negative consequences.

Insufficient requirement validation is also documented as a problem that exist in SRS. A sufficient validation of SRS is important to ensure that the requirements are completely and correctly specified the stakeholders' need [7]. Requirements are considered incomplete when important stakeholders' needs are not stated in the SRS. Misunderstanding between developer and customer may also lead to incorrect requirements specification [8]. During requirements gathering, problems may arise whenever a customer doesn't really know what they want. They only have vague idea of what they think they need [5]. Communication gap usually leads to misunderstanding, unclear and wrong interpretation [13].

Other problems related in SRS are:

- unreasonable timeline which usually arises from customer's demand. It is a common mistake for a developer to agree to such unreasonable timeline before actually performing a detail analysis.
- time-consuming testing and fixing defects process. Requirement engineering process is often inconsistently followed and significantly different from the as-documented requirements engineering method. As the result, inconsistent SRS are difficult to use and maintained[12].
- unprepared requirements engineers. People tend to think that anybody can write down requirements as they are using a native language. Without training, expertise or motivation, requirement engineers tend to face difficulties to understand and to follow good requirements methods which eventually leads to poor SRS [13].

Fig. 3 is a bar chart that illustrates the amount of papers (in percentage) where the main six problems in SRS can be found. These papers have been published from 2010 until 2015. Based on the chart, poor quality requirements have the highest number of papers, followed by natural language problem. This is then followed by insufficient requirement management, insufficient tool support, insufficient requirement verification and insufficient requirement validation.

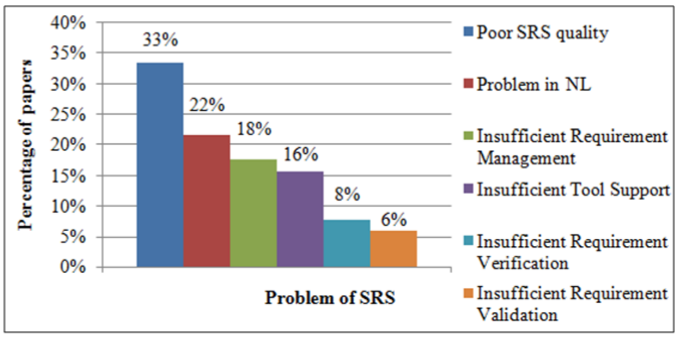

Fig. 3. The amount of research in SRS from 2010 to 2015

B. Result for Research Question 2: "How boilerplate contributes in improving SRS quality?

In order to answer this question, we reviewed literatures ranging from year 2000 to 2014 on boilerplate's contribution in improving SRS quality. According to the literature, boilerplate technique is proposed as a bridge between informal, NL specification and formal specification and was adopted as one of the semiformal language for the requirement specification in the CESAR [2]. Stenfan Farfeleder et al. use boilerplate in their tools that can semi-automatically transform NL requirements into semi-formal boilerplate requirements, called DODT. This tool is believed can reduce the manual effort of the transformation and to improve the quality of the requirements [52]. EARS boilerplate is one of the well-known boilerplate that have been use to overcome the NL problem. EARS provide five different requirement types translate into EARS template and reduce the ambiguity, vagueness and wordiness of the requirements [53)]. Furthermore, boilerplate is effective to reduce ambiguity and making them more amendable to automated analysis. It provides a simple and yet effective way for increasing the quality of requirements by avoiding complex structure, ambiguity and inconsistency in requirements [20].

Fig. 4 shows the coverage of boilerplate's research on improving quality attributes of SRS. According to the chart, disambiguity is a quality attribute where boilerplate contributes the most (30%), which is followed by Conformity (22%), Completeness, Accuracy and Reusability (13%) and other quality attributes (9%). The result suggests that, the quality problem in SRS is multidimensional and boilerplate technique has been found useful to improve SRS to overcome several quality problems. Nevertheless, these quality problems have been addressed individually, specific to the problems of concerned. Some other quality attributes (such as traceability and conciseness) that are important but yet have very limited coverage and remain unexplored to-date.

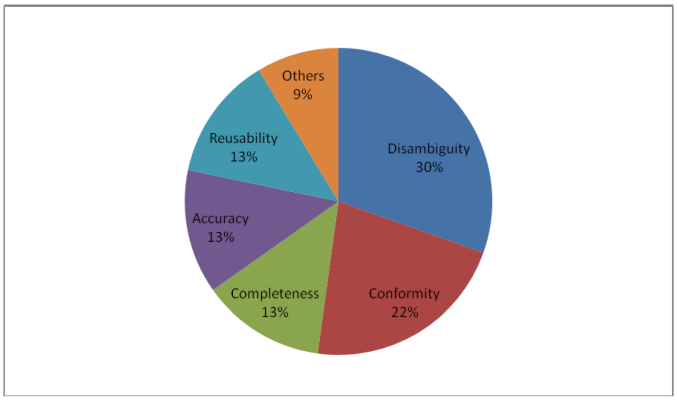

Fig. 4. Coverage of Boilerplate Contribution in Improving SRS Quality by Quality Attribute

IV. CONCLUSION

In order to ensure successful project completion, ensuring the quality of SRS is crucial. SRS quality is determined by examining several quality attributes such as accuracy, completeness and disambiguity. In this paper, we presented the results of SLR on the problems of SRS, with the focus on quality problems addressed by boilerplate technique. We discovered that, researchers work on six main SRS problems where the quality problem of SRS become the most popular of all. In addition, boilerplate technique's contribution covers multiple quality attributes where most of the work concentrate on disambiguity. In the future, we plan to explore boilerplate's contribution in other quality attributes that have limited research coverage but yet has significant impact on SRS's quality.

ACKNOWLEDGMENT

This research work is funded by Universiti Teknikal Malaysia Melaka (UTeM) Short Term Research Grant (PJP/203/FTMK (21B)/S1252). We gratefully thank our research groups Innovative Software System and Services (IS3) and Computational Intelligence Technologies (CIT) Lab (UTeM) for reviewing our work.

REFERENCES

[1] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, a. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, a. Ta, and M. Theofanos, "Identifying and measuring quality in a software requirements specification," *[1993] Proc. First Int. Softw. Metrics Symp.*, 1993.

[2] G. A. García-Mireles, M. Á. Moraga, F. García, and M. Piattini, "Approaches to promote product quality within software process improvement initiatives: a mapping study," *J. Syst. Softw.*, 2015.

[3] Q. a R. Cc, "Identifiing Requirement Conflicts," 1996.

[4] R. Rajnish and R. Vyas, "Writing Quality Requirments (SRS): An Approach To Manage Requirments Volatility," *Indian J. Comput. Sci. Eng.*, vol. 1, no. 1, pp. 28–37.

[5] D. Firesmith, "Common requirements problems, their negative consequences, and the industry best practices to help solve them," *J. Object Technol.*, vol. 6, no. 1, pp. 17–33, 2007.

[6] D. Firesmith, "Generating complete, unambiguous, and verifiable requirements from stories, scenarios, and use cases," *J. Object Technol.*, vol. 3, no. 10, pp. 27–39, 2004.

[7] A. T. Bahill and S. J. Henderson, "Requirements Development, Verification, and Validation exhibited in famous failures," *Syst. Eng.*, vol. 8, no. 1, pp. 1–14, 2005.

[8] D. E. Damian and D. Zowghi, "The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization," *Proc. IEEE Jt. Int. Conf. Requir. Eng.*, no. July 2001, pp. 1–10, 2002.

[9] A. Pflüger, W. Golubski, and S. Queins, "Tool-supported model-driven validation process for system architectures," *Proc. 5th Int. Work. Model Based Archit. Constr. Embed. Syst. - ACES-MB '12*, pp. 1–6, 2012

[10] A. Batool, Y. H. Motla, B. Hamid, S. Asghar, M. Riaz, M. Mukhtar, and M. Ahmed, "Comparative study of traditional requirement engineering and Agile requirement engineering," *Adv. Commun. Technol. (ICACT), 2013 15th Int. Conf.*, pp. 1006–1014, 2013.

[11] A. R. da Silva, "Quality of requirements specifications," *Proc. 29th Annu. ACM Symp. Appl. Comput. - SAC '14*, pp. 1021–1022, 2014.

[12] E. Approach, Raimundas Matulevicius Process Support for Requirements Engineering, no. June. 2005.

[13] J. Coughlan and R. D. Macredie, "Effective Communication in Requirements Elicitation: A Comparison of Methodologies," *Requir. Eng.*, vol. 7, no. 2, pp. 47–60, 2014.

[14] A. Bauer, M. Leucker, and C. Schallhart, "Runtime Verification for LTL and TLTL," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 4, pp. 1–64, 2011.

[15] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Requirement Boilerplates : Transition from Manually-Enforced to Automatically-Verifiable Natural Language Patterns," pp. 1–8, 2014.

[16] M. Heindl, F. Reinisch, S. Biffl, A. Egyed, and M. Del Rey, "Value-Based Selection of Requirements Engineering Tool Support," *Ieee*, 2006.

[17] H. E. Smith, "the Language of Property: Form, Context, and Audience," *Stanford Law Rev.*, vol. 1105, pp. 1–78, 2003.

[18] U. C. Berkeley, "Law and Economics Workshop UC Berkeley," 2004.

[19] V. Ambriola and V. Gervasi, "Processing natural language requirements," *Proc. IEEE Int. Autom. Softw. Eng. Conf. ASE*, pp. 36–45, 1997

[20] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Requirement Boilerplates : Transition from Manually-Enforced to Automatically-Verifiable Natural Language Patterns," pp. 1–8, 2014

[21] N. L. Requirements, "Chapter 2 AMBIGUITY IN REQUIREMENTS SPECIFICATION," 2004.

[22] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, and R. Gnaga, "Automatic checking of conformance to requirement boilerplates via text chunking: An industrial case study," *Int. Symp. Empir. Softw. Eng. Meas.*, pp. 35–44, 2013.

[23] C. Arora, M. Sabetzadeh, L. C. Briand, F. Zimmer and Gnaga, R. (2013, August). RUBRIC: A flexible tool for automated checking of conformance to requirement boilerplates. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering* (pp. 599-602). ACM

[24] M. Dorfman, "Requirements Engineering," pp. 1–30, 1999.

[25] S. Lauesen and O. Vinter, "Preventing Requirement Defects: An Experiment in Process Improvement," *Requir. Eng.*, vol. 6, no. 1, pp. 37–50, 2001.

[26] O. Daramola, G. Sindre, and T. Stalhane, "Pattern-Based Security Requirements Specification Using Ontologies and Boilerplates," pp. 54–59, 2012.

[27] I. L. Margarido, J. P. Faria, R. M. Vidal, and M. Vieira, "Classification of defect types in requirements specifications: Literature review, proposal and assessment," *6th Iber. Conf. Inf. Syst. Technol. (CISTI 2011)*, no. 1, pp. 1–6, 2011. Butler, R.E., & Mitchell, T.A. 2012. 'Method and system for software defect management': Google Patents.

[28] O. Park, "(12) United States Patent US . Patent Capturing software artifacts into databases Managing a test Identifying a test case for," vol. 1, no. 12, 2012.

[29] J. H. Hayes, "Building a requirement fault taxonomy: experiences from a NASA verification and validation research project," *14th Int. Symp. Softw. Reliab. Eng. 2003. ISSRE 2003.*, 2003.

[30] G. S. Walia and J. C. Carver, "A systematic literature review to identify and classify software requirement errors," *Inf. Softw. Technol.*, vol. 51, no. 7, pp. 1087–1109, 2009.

[31] M. Kalinowski, E. Mendes, D. N. Card, and G. H. Travassos, "Applying DPPI: A Defect Causal Analysis Approach Using Bayesian Networks," vol. 6156 LNCS, pp. 92–106, 2010.

[32] G. S. Walia and J. C. Carver, "Evaluating the use of requirement error abstraction and classification method for preventing errors during artifact creation: A feasibility study," *Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE*, pp. 81–90, 2010.

[33] G. Kotonya and I.Sommerville, Requirement Engineering Process and Techniques. West Sussex,England. John Wiley & Sons Ltd, 1998.

[34] Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. Software Engineering, IEEE Transactions on, 37(4), 509-525.

[35] B.A. Kitchenham and S. Charters, "Procedures for Performing Systematic Literature Review in Software Engineering," EBSE Technical Report version 2.3, EBSE-2007.

[36] Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. Information and software technology, 51(1), 7-15.

[37] Proynova, R., Paech, B., Wicht, A., & Wetter, T. (2010). Use of personal values in requirements engineering–A research preview. In Requirements Engineering: Foundation for Software Quality (pp. 17-22). Springer Berlin Heidelberg.

[38] Knauss, E., Brill, O., Kitzmann, I., &Flohr, T. (2009, May). Smartwiki: Support for high-quality requirements engineering in a collaborative setting. In Wikis for Software Engineering, 2009. WIKIS4SE'09. ICSE Workshop on (pp. 25-35). IEEE.

[39] E. Juergens, F. Deissenboeck, M. Feilkas, B. Hummel, B. Schaetz, S. Wagner, C. Domann, and J. Streit, "Can clone detection support quality assessments of requirements specifications?," *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. - ICSE '10*, vol. 2, p. 79, 2010.

[40] N. Salleh, E. Mendes, and J. Grundy, "Empirical Studies of Pair Programming for CS / SE Teaching in Higher Education : A Systematic Literature Review," 2010.

[41] P. G. Maropoulos and D. Ceglarek, "Design verification and validation in product lifecycle," *CIRP Ann. - Manuf. Technol.*, vol. 59, no. 2, pp. 740–759, 2010.

[42] K. Pohl, "The three dimensions of requirements engineering," *Adv. Inf. Syst. Eng.*, vol. 6353, pp. 275–292, 1993.

[43] S. Lawrence, "Software qualit-y-:," no. January, 1996.

[44] N. A. Rosmadi, S. Ahmad, and N. Abdullah, "The Relevance of Software Requirements Defects Management to Improve Requirements and Product Quality : A Systematic Literature Review."

[45] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated analysis of requirement specifications," *Proc. - Int. Conf. Softw. Eng.*, pp. 161–171, 1997.

[46] O. C. Gotel, A. C. W. Finkelstein, and L. Sw, "An Analysis of the Requirements Traceability Problem Imperial College of Science , Technology & Medicine Department of Computing , 180 Queen ' s Gate," pp. 94–101, 1994.

[47] Wall, G. W. & Whalen, N. 2001, 'The Boiletplate: What Does It Accomplish?'.

[48] Farfeleder, S., Moser, T, Krall, A., Stalhane, T., Zojer, H. & Panis, C.2011, 'DODT: Increasing Requirements Formalism using Domain Ontologies for Improved Embedded Systems Development'.

[49] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," *ICSE '00 Proc. Conf. Futur. Softw. Eng.*, vol. 1, pp. 35–46, 2000.

[50] S. J. Choi and G. M. Gulati, "Innovation in Boilerplate Contracts: an Empirical Examination of Sovereign Bonds.," *Emory Law J.*, vol. 53, no. 3, pp. 929–996, 2004.

[51] T. Wien, F. Reichenbach, E. Carlson, and T. Stålhane, "Reducing development costs in industrial safety projects with CESAR," *Proc. 15th IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA 2010*, no. i, 2010.

[52] S. Farfeleder, T. Moser, A. Krall, H. Zojer, and C. Panis, "DODT : Increasing Requirements Formalism using Domain Ontologies for Improved Embedded Systems Development," 2011.

[53] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "EARS (Easy Approach to Requirements Syntax)," *Proc. IEEE Int. Conf. Requir. Eng.*, pp. 317–322, 2009.

[54] Mott, Frank Luther (1957). *A History of American Magazines, v.4*. Cambridge(MA): The Belknap Press of the Harvard University Press. pp. 53–54.

[55] B. Scott, "RECONSIDERING BOILERPLATE : CONFRONTING NORMATIVE AND DEMOCRATIC DEGRADATION Most of us are used to receiving paperwork (or its electronic equivalent) during transactions . We are given forms to sign when we rent an automobile or an apartment , and pile," 2012.

[56] S. Farfeleder, T. Moser, A. Krall, T. Ståalhane, I. Omoronyia, and H. Zojer, "Ontology-driven guidance for requirements elicitation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6643 LNCS, no. PART 2, pp. 212–226, 2011.

[57] K. Pohl and C. Rupp, Requirements Engineering Fundamentals, 1st ed. Rocky Nook, 2011.

Appendix

Search Focus	Description	
"Unambiguous"	Focus	How to reduce ambiguity.
	Approach	Approaches that have been done to reduce ambiguity.
"Complete"	Focus	How to reduce incompleteness.
	Approach	Approaches that have been done to reduce incompleteness.
"Correct"	Focus	How to reduce incorrectness.
	Approach	Approaches that have been done to reduce incorrectness.
"Understandable"	Focus	How to reduce non-understandability.
	Approach	Approaches that have been done to reduce non-understandability.
"Verifiable	Focus	How to reduce unverifiable.
	Approach	Approaches that have been done to reduce unverifiable.
"Consistent"	Focus	How to reduce inconsistency.
	Approach	Approaches that have been done to reduce inconsistency.
"Achievable"	Focus	How to avoid unattainable requirement.
	Approach	Approaches that have been done to avoid unattainable requirement.
"Concise"	Focus	How to reduce wordy SRS.
	Approach	Approaches that have been done to reduce wordy SRS.
"Design Independent"	Focus	How to achieved design independent.
	Approach	Approaches that have been done to improve design independent SRS.
"Traceable"	Focus	How to reduce untraced requirement.
	Approach	Approaches that have been done to reduce untraced requirement.
"Modifiable"	Focus	How to achieved modifiability.
	Approach	Approaches that have been done to improve modifiable.
"Interpretable"	Focus	How to achieved interpretability.
	Approach	Approaches that have been done to improved interpretable.
"Stability"	Focus	How to reduce achieved stability.
	Approach	Approaches that have been done to improve stability.
"Not redundant"	Focus	How to reduce redundancy.
	Approach	Approaches that have been done to reduce redundancy.
"Precise"	Focus	How to make precise requirement.
	Approach	Approaches that have been done to improve precise SRS.
"Reusable"	Focus	How to increase reusability.
	Approach	Approaches that have been done to improve reusability.
"Organized"	Focus	The advantages of organized SRS.
	Approach	Approaches that have been done to improve organized SRS.