Journal of Theoretical and Applied Information Technology
31st December 2024. Vol.102. No. 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCING CLARITY IN CHINESE SOFTWARE REQUIREMENTS: A BOILERPLATE-BASED APPROACH TO IMPROVE REQUIREMENT EXPRESSION

HE JIAYING[1,2], MOHD HAFEEZ OSMAN[1,*], SA'ADAH HASSAN[1], NG KENG YAP[1,*]

[1] Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia

[2] Faculty of Physics & Information Engineering, Zhaotong University, Zhaotong, Yunnan, China

Email : hjy_cityu@163.com, hafeez@upm.edu.my, saadah@upm.edu.my, kengyap@upm.edu.my

ABSTRACT

During the rapid process of software development, requirements engineering plays a crucial role in ensuring the success of software projects. Clear and accurate requirements specification directly impact the effectiveness of project implementation, influences customer satisfaction, and ultimately determines the overall success rate of the project. However, ambiguity and structural issues are common in Chinese software requirements, which can hinder understanding and implementation. These issues not only increase the complexity of project execution but may also lead to extended development cycles and wasted resources. To address these challenges, this paper aims to develop a set of Chinese-based software requirement boilerplate to improve the clarity and accuracy of requirements specification. The boilerplate is designed with consideration for Chinese grammar conventions and international standards, offering a universal and user-friendly solution. This paper reviews related literature, discuss the limitations of current requirements specification method, and details the boilerplate design process and evaluation methodology. The effectiveness of the boilerplate is demonstrated through the reconstruction of requirements statements across various software industries, complemented by expert assessments and feedback from industry professionals. This study provides a practical foundation for Chinese software requirements specification, aiming to enhance the quality and efficiency of requirements expression.

Keywords: *Software Requirements Engineering, Chinese Requirements Expression, boilerplate Design, Clarity, Requirements Normalization*

1. INTRODUCTION

In software development, Software Requirements Engineering (SRE) is crucial for project success. Unclear requirements, frequent changes, and validation challenges can lead to delays, budget overruns, or project failure. The primary methods for expressing requirements are formal, informal, and semi-formal descriptions. Formal methods use mathematical or logical language for verifiable and unambiguous requirements, suitable for high-security and stability systems. Informal methods utilize natural language, which is easy to

understand but can be ambiguous and should be used cautiously. Semi-formal methods blend the benefits of both, offering a balance of accuracy and flexibility, making them suitable for requirements documents and user stories[1][2][3].

To address the issues with software requirements in the Chinese software development industry, we conducted a questionnaire survey with 422 practitioners from 170 companies. The results reveal that over 77% of Chinese software requirements are expressed in natural language, which often leads to ambiguity and comprehension bias. Among respondents, 21.28% reported "always encountering ambiguous requirements," 30.5% "often encounter them," and 41.13% "generally encounter them." The survey indicates that ambiguous requirements directly contribute to project failure. The ambiguity in Chinese language spans multiple levels, including phonology, vocabulary, syntax, semantics, and context[4].

The emergence of ambiguities in Chinese mainly arises from the language's complexity and diversity, which can lead to comprehension bias. Key causes include the trend of language simplification, contextual diversity, and the structural characteristics of Chinese. This ambiguity highlights a fundamental contradiction in human communication: clear and accurate expression requires more linguistic elements, while people often favor simpler, more common language, resulting in various ambiguous phenomena. Thus, addressing the ambiguity in Chinese software requirement descriptions is of great significance.

Therefore, there is an urgent need for an effective model to improve the quality of Chinese software requirements. Boilerplate, as an informal software requirement specification method, was first created for the American printing industry and later used to refer to textual requirement templates in requirements engineering. Its core component is a natural language model, which enhances the automated analysis of natural language requirement statements by restricting the syntactic structure of requirement phrases, and reduces the ambiguity of natural language requirement statements[5]. A number of standardized requirements boilerplate have been proposed internationally, e.g. EARS [6], Pohl and Rupp [7], which provide specifications for the preparation of high-quality requirements documents. However, a natural language-based boilerplate for Chinese software requirements has not yet appeared. To solve this problem, this study aims to construct a standardized boilerplate specifically for Chinese software requirements. The boilerplate not only standardizes the way Chinese software requirements are written, but also improves verifiability and communication efficiency by reducing ambiguity and inconsistency in the requirements. This study combines the Delphi method of aggregating and optimizing expert opinions to ensure the versatility and practicability of the template, and verify the effectiveness of the boilerplate in real projects to provide a reference for the expression of software requirements in the future.

The contributions of this paper is listed below:

(1) The study provides a summary of the categories of Chinese software requirements and the classifications of Chinese ambiguity.

(2) Based on literature review and real-world examples, this study develops a series of Chinese software requirement boilerplate that adhere to both Chinese grammatical rules and international standards, aiming to enhance the quality of Chinese requirements by improving clarity, completeness,

singularity, verifiability, conformity, and understandability.

(3) The structure of this paper is listed below: Section 2 provides the background study, reviewing relevant literature and previous research in the field. Section 3 outlines the methodology, detailing the approach and techniques employed in the study. Section 4 presents the Chinese software requirements boilerplate, explaining its formulation, as well as the features, structure, and categories of the boilerplate. Section 5 reports the results of the research, examining the normative aspects of the software requirements boilerplate. Finally, Section 6 presents the discussion, highlighting the advantages of applying the boilerplate and comparing our work with relevant studies.

2. BACKGROUND STUDY

Software Requirements Engineering (SRE) has been widely researched and paid attention to as a key part of the software development life cycle. One of the main reasons for software project failure is unclear requirements definition or problematic requirements documentation. In order to improve the quality of requirements, researchers have proposed a variety of methods and tools, especially in the design and application of requirements boilerplate, which have made significant progress.

2.1 The effects of boilerplate on software requirements quality

One of the primary factors contributing to the failure of software projects is the inadequate definition of requirements or flawed requirements documentation, as indicated in references [8] and [9]. Numerous scholarly sources advocate that the implementation of standardized requirements boilerplate can significantly enhance the quality of requirements documentation. Studies conducted by Barbosa, Cerqueira, and Da Cunha (2023), Tian, Yin, and Xiao (2022), Großer, Rukavitsyna, and Jürjens (2023) propose that the utilization of standardized boilerplate not only mitigates ambiguity in document creation but also fosters enhanced collaboration and communication among teams, as cited in references [10], [11], and [12] respectively. These studies emphasize that through the constraints imposed by boilerplate, development teams can capture requirements in a more systematic manner, thereby ensuring their completeness and consistency. Furthermore, Ramesh and Reddy (2021), in reference [13], underscore that the use of requirements boilerplate enhances the verifiability of requirements documents, which subsequently leads to more efficient reviews and testing of requirements. A clear and structured presentation of requirements diminishes the frequency of requirement changes within a project, ultimately elevating the project's success rate.

2.2 The criteria of boilerplate

IEEE 830, ISO/IEC 29148 and GB-T9385-2008 are international and Chinese standards for Software Requirements Specification (SRS) to guide the preparation and management of requirements documents[14][15][16]. They provide definitions for key software requirements, including the characteristics of being unambiguous, complete, singular, feasible, verifiable, correct, conforming, necessary, appropriate, consistent, comprehensible, validated, ranked for importance, modifiable, and traceable[14][15][16].

2.3 Available Boilerplate

software requirement boilerplate vary across languages. English-based boilerplate, such as EARS, adv-EARS, RUPP, and ISO/IEC/IEEE 29148:2018, use restricted natural language to express requirements [5][7][14][17]. Some boilerplate, like RUPP, have been translated into other languages, including Turkish, German, Thai and Spanish, to align with the grammatical structures of those languages [18][19][20][21]. Additionally, there are boilerplate tailored to specific linguistic features,

such as the IsiZulu boilerplate, which focuses on sentence structure and quantitative expressions [14] .

Boilerplate types including Ubiquitous, Event-Driven, Unwanted Behavior, State-Driven, Optional Features, and others.

Boilerplate structures encompass Sentence Structure, Requirement Description, Conditional boilerplate, and various others. Quantification types include Universal, Existential, Subsumption and Negative Subject Concord Quantification. Descriptions cover conditional, general form, attribute, quantity, location, numeric value and verbal descriptions.

2.4 Chinese Language Ambiguity

In the English-speaking world, ambiguity is often linked to vagueness, uncertainty, and multiple meanings in language. ISO/IEC/IEEE 29148:2018 specifies that extreme terms (e.g., "best"), subjective expressions (e.g., "user friendly"), vague pronouns (e.g., "it"), logical connectors (e.g., "or"), and generalized terms (e.g., "all" or "most") should be avoided, as they can introduce ambiguity and complicate the interpretation and validation of requirements [22]. IEEE 830 and GB-T 9385-2008 also stress the importance of using clear, unambiguous terms and suggest clarifying polysemous words in a glossary[15][23] .

In Chinese, ambiguity exists at multiple levels, including phonology, vocabulary, syntax, semantics, and pragmatics. Phonological ambiguity arises from homophones and variations in stress, while lexical ambiguity stems from words with multiple meanings. Grammatical ambiguity is caused by flexible word structure and differences in word modification scope. Semantic ambiguity occurs due to differences in meaning or omission, and contextual ambiguity results from varying interpretations in different cultural or situational contexts [24][25][26].

Ambiguity can be further categorized into lexical and syntactic types. Lexical ambiguities including word meaning confusion, polysemy, quantifier ambiguity, and comparative ambiguity [23][26][27][28][29]. Syntactic ambiguities, such as passive voice, hierarchical structures, and attachment issues, arise from unclear sentence structures and grammatical choices[32].

2.5 Chinese Software Requirements Expression Challenges

Despite progress in software requirements standardization, research on expressing of requirements in Chinese environments is still in its early stages. In Chinese requirement documents, misinterpretations between requirement personnel and developers often lead to deviations in later development stages. Current research mainly addresses linguistic issues, with limited systematic work on developing standardized boilerplate for Chinese language features. Shen (2024) notes that the flexibility of Chinese makes descriptions more concise but can lead to ambiguity, suggesting that grammatical boilerplate could improve clarity while maintaining brevity [30]. The diversity of Chinese vocabulary and grammar also complicates automated validation of requirements. While the General Administration of Quality Supervision, Inspection and Quarantine of China (AQSIQ) and the National Standardization Administration of China (2008) have provided guidelines for writing requirements in Chinese [15]. However, a standardized format for requirement statements is still lacking, limiting the effectiveness of these guidelines in real projects.

2.6 Boilerplate and their testing methods

This study involves designing, validating, and evaluating a standardized boilerplate for Chinese software requirements. Currently, there is no

systematic method for assessing requirement boilerplate[31]. Evaluations are often qualitative or quantitative, typically conducted by practitioners. For example, Oztekin and Dalveren (2023) created a Turkish boilerplate based on RUPP and had requirements engineers revise existing requirements using it[5]. Großer et al. (2023) similarly invited engineers to assess requirements before and after using their boilerplate[11]. The Delphi method, used by Thangaratinam and Redman (2005), and Beiderbeck et al. (2021), gathers expert feedback to refine boilerplate over multiple rounds [32][33] .

This study aims to address the lack of language specification in Chinese software requirements by proposing a standardized boilerplate. The Delphi method gathers feedback from experts in both Chinese language and software requirements, evaluating the boilerplate's syntax, clarity, and alignment with international standards like IEEE 830, ISO/IEC/IEEE 29148:2018, and GB/T 9385-2008.

3. METHODOLOGY

This study employs a multi-step methodology to create a standardized template for Chinese software requirements expression, utilizing literature analysis and expert interviews based on the Delphi method. These methods ensure the boilerplate's validity, feasibility, and applicability. The research framework is shown in Figure 1.

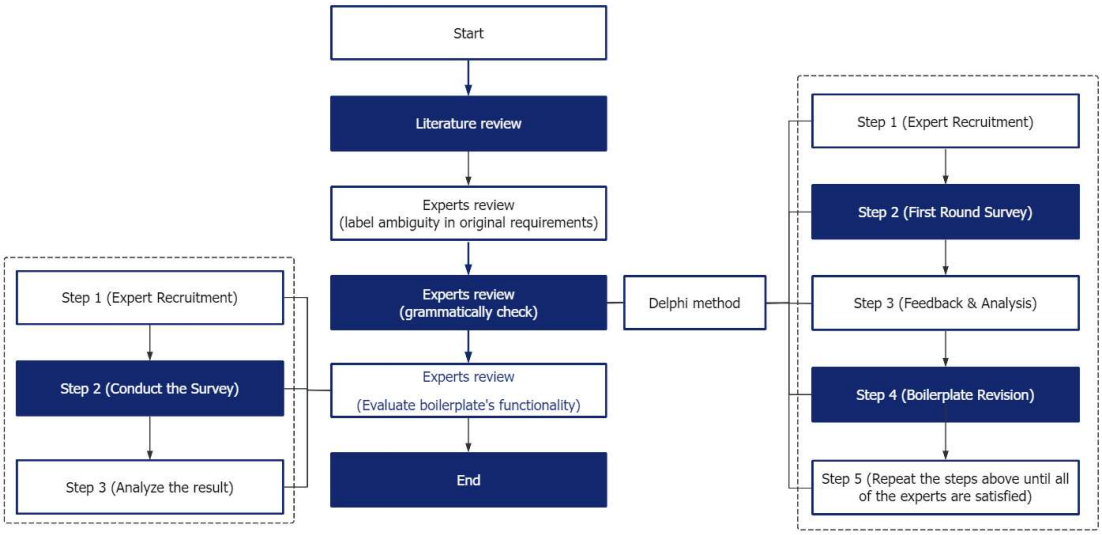

Figure 1. Research framework

3.1 Literature review techniques

By conducting literature review, this study summarizes aspects including international standards for evaluating software requirement boilerplate, the types of currently available boilerplate, the sentence components included in their presentation, as well as the types of Chinese ambiguities and their underlying causes.

3.2 Expert Review: Labeling Ambiguities in Requirements

Seven senior software engineering experts, each with over five years of experience, were invited to label the ambiguity in 3,697 Chinese software requirement statements independently. These requirements were gathered from industries. The experts review focused on annotating the original

requirements to identify and classify ambiguous instances.

3.3 Expert Review: Grammatical Accuracy Check

To ensure the grammatical accuracy and comprehensibility of the Chinese software requirement boilerplate, we invited Chinese language experts to conduct five rounds of rigorous review using the Delphi method. Firstly, based on the annotations made by seven senior software engineering experts on 3,697 Chinese requirement statements, non-ambiguous statements were filtered and analyzed. Drawing on the linguistic elements and classifications contained in the international boilerplate, the types of Chinese software requirement boilerplate were summarized, and the initial structure of the software requirement syntax boilerplate was designed for each type. Subsequently, the original Chinese requirement statements were rewritten using the current preliminary boilerplate obtained for software requirement description statements, a Malaysian Chinese lecturer and a Singaporean Chinese NLP researcher conducted pre-tests, five senior Chinese experts (including two university professors, two university lecturers, and a secondary school teacher with 30 years of experience) reviewed the boilerplate in multiple rounds, and the research team optimized the boilerplate based on the experts' feedbacks.

3.4 Expert Review: Evaluate the functionality of the boilerplate

This study uses quantitative analysis to evaluate the performance of software requirement boilerplate. The evaluators should be senior software engineers, and the evaluation standards include IEEE 830, ISO/IEC/IEEE 29148:2018, and GB/T 9385-2008. Among the criteria within these standards, traceability, modifiability and importance ranking of the boilerplate can be improved by tools such as JIRA, Rational DOORS and Git, while other aspects (e.g., correctness and feasibility) are assessed manually by software engineers. Therefore, this study only assesses the changes in non-ambiguity, completeness, singularity, verifiability, consistency and understandability of software requirement boilerplate by mixing the original requirement documents with the boilerplate rewritten requirement documents. The experiment hypothesizes that the rewritten requirements are better in the evaluated criteria.

4. THE CHINESE SOFTWARE REQUIREMENTS BOILERPLATE

This section explains the boilerplate for Chinese software requirements which include how the boilerplate is formulated and the features, structures and categories of the boilerplate.

4.1 Features, Structures, and Categories of Existing Boilerplate

From the literature review, it was identified that existing boilerplate adopt a modular design approach, encompassing core components such as subject (subject, predicate, complement) and hierarchical divisions (e.g., system structure, states, events, activities, temporal constraints, and performance metrics). These boilerplate utilize placeholders (e.g., `<entity>`, `<property>`, `<value>`, `<comparison>`) to ensure consistency, precision, and flexibility of expression, thereby effectively minimizing ambiguity. The hierarchical framework supports the description of static information (e.g., composition, arrangement, attribute states), dynamic behaviors (e.g., events, activities), temporal relationships (e.g., timing constraints, sequential dependencies), performance metrics (e.g., rate, capacity), and logical inferences (e.g., conditions, possibilities, decision-making). Furthermore, the boilerplate exhibit versatility by covering various types, including entity descriptions,

behavioral specifications, attribute constraints, temporal conditions, event-driven logic, performance requirements, aggregate operations, and nested structures.

4.2 The Structure and Types of Chinese Software Requirements Boilerplate

The Chinese software requirement boilerplate summarized by the research team follow a modular structure that describes specific conditions, behaviors and operations through prefixes, bodies and suffixes. As for the boilerplate's format, contained within 【】 is considered as a whole, / separates multiple choices. Requirement types define the kinds of requirements, each requirement type and version (e.g., conditional V1, state-driven V2) is designed to handle specific scenarios and operations. The prefix usually contains the system location or state that provides the contextual environment for the requirement. The main body describes the system components, system behaviors and user actions involved and is supported by placeholders (e.g., <system location>, <system component>, <user behavior>, etc.) to ensure flexibility of the boilerplate. Suffixes provide further context or conditions, such as temporal constraints (e.g., 'after', 'when') or optional conditions (e.g., 'optional', 'default'), enabling the boilerplate to be used to satisfy different circumstances.

4.3 Chinese requirement boilerplate

The study classifies Chinese software requirement statements into eight categories based on a review of relevant literature and annotations from seven senior software engineering experts (TABLE 1). These categories include Conditional Requirements (actions triggered by specific conditions), State-driven Requirements (responses to state changes at certain locations), Autonomous Requirements (automated actions based on time or default conditions), User Interaction Requirements (actions triggered by user input), Interface Requirements (automatic actions following state changes), Optional Functional Requirements (functions users can enable or disable), Functional Inclusion Requirements (boundaries that must be met by the system), and Complex Requirements (requirements combining multiple patterns and conditions).

In addition, three types of supplementary descriptive statements are introduced to provide further context. Conceptual Descriptions define the fundamental characteristics and relationships of an object or concept, Location Descriptions specify the location or spatial relationship of an object, and Attribute Descriptions describe the features or attributes of an object. These descriptive statements complement the main requirement categories and enhance the clarity of the boilerplate, which can be seen in TABLE 2.

Table 1: The boilerplate for Chinese software requirements

Requirement type	prefix	Body	Suffix
conditional (V1)	<System location> / *n <System status>	如(果)/ 只要 if / as long as	那么 / 则 then
		*n 【<System component><System component status>】,	*n 【<System Components> (optional) 就 / 会 at once (optional)<System behavior>】
conditional (V2)	<System location> / *n <System status> <System location> / *n	只有 / 需要 Only/Required	<executive subject> (optional) *n 才可 can /will（optional） <User behavior><System component><Action description>
		【 <User behavior><action object> 】*n / <System component><System component status>】	
		时/后, when/after,	
State-Driven (V1)	<System status>	<System component >随着/跟/与/跟随/根据 follow <System status>而/进行 while/proceed (optional)	<system behavior><Action description>
State-Driven (V2)	<System location> / *n		有 has <System component>
State-Driven (V3)	<System status>	【<System component ><system behavior>】 / 【 <interaction behavior><interaction element> 】后 then, / 并 / 且 and	<System behavior><System Components>(optional) *n
Autonomous (V1)	<System location> / *n	<System component > 默认/自动/定时 will automatically/timely	<System behavior><System Components>(optional) *n
Autonomous (V2)	<System status>	<System component >（optional） 默认 by default(optional) 有/是/在/显示 have/be/at/show	<System component ><System component description>(optional)
Autonomous (V3)	<System location> / *n	<System component >（optional）	每 every <duration><System behavior> 一次 once, <action description>（optional）
Autonomous (V4)	<System status>	<System behavior>	为<duration >一次
User-Interaction (V1)	<System location> / *n <System status>	<User behavior><System Components> *n	<System behavior> *n <System Components>(optional)
		时 / 后 when/then (optional)	
User-Interaction (V2)	<System location> / *n <System status>	*n 将 <system properties ><User behavior> 到 <System Components>	<System behavior> *n <System Components>(optional)
		时 / 后 when/then (optional)	
Interface	<System location> / *n <System status>	<System Components/Users> (optional) *n <System behavior> <System Components> (optional),	<System behavior> *n <System Components>(optional)
		时/之后/后， when/ after,(optional)	
Optional Function	<System location> / *n <System status>	<user>(optional) 可 can	<System Components> (optional) *n 将/可 will (optional) <System behavior> <system properties >(optional)
		<User behavior> *n <System Components>	
Functional Containment(V1)	<System location> / *n	<System Components>	*n 不 / 不可 cannot <System behavior> <system properties >(optional)
Functional Containment(V2)	<System status>		只 only <System behavior> *n <System Components>(optional)
Functional Containment(V3)	<System location> / *n <System status>	<System Components>	为 is / 无 have no *n <system properties >

Table 2: System action descriptions

Concept description	<System Components> 包括 / 见 / 是 Include / can be seen on / be <System Components>
Location description	<system properties > 位于 <System Components> <direction position> <System Components><direction position>是 is<System Components>
Attribute description	<system properties > ： list of <system properties > <system properties > 跟/ 同 / 在 with <system properties > (optional)<state description>(optional) <System Components>(optional)<System behavior><specific content > <System Components><System behavior><system properties > <System Components><System behavior> <System Components><System behavior>(<explanation>)

5. RESULTS

This section will examine the normative aspects of the software requirements boilerplate. It will first introduce the respondents' identities and professional backgrounds, then assess the reliability of the questionnaire results. Finally, it will detail the improvements the boilerplate brings to each metric of the software requirement description. This analysis will provide a basis for evaluating the boilerplate's practical effectiveness.

The quality of the software requirements boilerplate was assessed by eleven software practitioners, consisting of two software developers, three project managers, four requirements analysts, and two testers. Of these, 90.9% (10) had over ten years of experience in requirements engineering, 9.09% (1) had 6-10 years of experience, and all had worked on five or more software projects. Among them, 72.73% (8) primarily used natural language for requirements expression, 18.18% (2) used UML, and 9.09% (1) used decision tables. The pre-tests for this section were carried out by two product managers with nine years of experience working in Shenzhen, China's first-tier city.

Randomly mixing the original and rewritten statements helps eliminate investigator bias, ensuring objectivity and fairness in the evaluation. This approach reduces the influence of subjective judgment on the results. Each software requirement was assessed across six dimensions: Unambiguous, Complete, Singular, Verifiable, Conforming, and Comprehensible.

For validity testing, the Kaiser-Meyer-Olkin (KMO) method was used, with a score of 0.60 or higher considered acceptable[34]. To assess item consistency, Cronbach's alpha was calculated, with values above 0.70 is considered reliable[35]. The KMO values ranged from 0.84 to 0.90, and Cronbach's alpha values ranged from 0.972 to 0.978, indicating strong validity and reliability of the survey items for evaluating the developed interface, as shown in Table 3.

Table 3: KMO and Cronbach's alpha test outcomes

Question	KMO	Cronbach's alpha
clarity	0.85	0.974
completeness	0.88	0.976
singularity	0.87	0.975
Verifiability	0.84	0.973
Conformity	0.90	0.978
Understandability	0.86	0.972

After rewriting, the software requirement use cases showed improvements in Unambiguous, Complete, Singular, Verifiable, Conforming, and

Comprehensible. Notably, the most significant improvements were observed in Unambiguous and Singular, followed by Complete, Verifiable, Conforming, and Comprehensible. Table 4 presents a comparison of evaluation metrics for software requirements before and after revision.

Table 4: Comparison of evaluation metrics for software requirements before and after improvement

Indicators	Before	After
clarity	2.1	3.3
completeness	2.1	3.2
singularity	2.1	3.3
Verifiability	2.2	3.3
Conformity	2.2	3.2
Understandability	2.2	3.3

6. DISCUSSION

In the process of describing software requirement statements, rigorous presentation and accurate analysis are crucial, as any errors may have a negative impact on the development of subsequent programs. Standardized templates can effectively improve the clarity and accuracy of requirement statements and reduce ambiguities caused by the flexibility of Chinese presentation.

If manual checking of software requirements is adopted, it usually requires a lot of human and material resources, especially in complex software engineering projects, involving the collaboration of multiple stakeholders, where it is more challenging to maintain consistency in requirements descriptions. Therefore, it is particularly important to adopt an automated or simplified approach to requirements rewriting and assessment. This not only helps to ensure the uniformity of software requirement descriptions, but also effectively reduces ambiguities, which enhances team members' understanding and promotes efficient collaboration.

This study demonstrates the process of constructing a standardized software requirements template, which involves a variety of methods such as literature review, case study, and expert assessment. The experimental results show that the template achieves significant results in reducing statement ambiguity and enhancing requirements reusability, consistency, completeness, singularity, verifiability and understandability.

Table 5: Center Table Captions Above The Tables

Study	Method/Tool	Evaluation Method	Focused domain
Oztekin & Dalveren (2023) [36]	Developed a Turkish boilerpla based on RUPP	Qualitative evaluation from domain-specific requirements engineers.	e-Government
Großer, Rukavitsyn and Jürjens (2023) [	boilerplate comparison	quality guidelines	"Master" is the most useful boilerpl
Antoniou & Bassilia (2024) [38]	Developed a tool using ontolo based on the natural language library (Resource Description Framework triples).	Case study and user assessments.	ATM

Großer, Rukavitsyna and Jürjens (2024) [	compare between free-text requirements and treatment groups rewrote with different boilerplates	quality guidelines	boilerplates can generally improve quality
de Mello Barbosa, Cerqueira, & da Cur (2023) [40]	compare between 54 natural language boilerplates	quality guidelines	cross-industry
Our Work	Developed a Chinese natural language boilerplate checked b Chinese linguists and software engineers manually	quality guidelines.	cross-industry

Our work stands out for its use of natural language, the most widely used method for expressing requirements, offering a flexible and generalizable solution across industries, unlike domain-specific approaches in other studies. Besides, the focus on Chinese software requirements boilerplate is a significant contribution. Additionally, our work paves the way for leveraging advanced Chinese AI technologies to automate the handling of Chinese software requirements. We build on existing studies. For example, Bao et al. (2021) introduced the RNL2SysML method, which automatically generates SysML models from restricted Chinese requirements, improving accuracy and efficiency [41]. Chen et al.'s (2023) NLP-based approach for converting free-text requirements into SysML models [42], Wang et al.'s (2023) syntax correction method [43], Xu and Wang's (2022) word segmentation and POS tagging model [44], Chai and Wang's (2022) enhanced named entity recognition using BERT and BILSTM [45], and Zhu et al.'s (2022) TF-IDF-based K-Means clustering for requirement classification [46]. These advancements all contribute to improving the precision of model generation. However, these technologies still require the development of a standardized set of Chinese natural language templates to address ambiguity in requirement expressions.

Potential limitations of the study include sample selection bias, assessment subjectivity, survey design issues, external factors, boilerplate applicability and statistical analysis limitations. Assessors' backgrounds may not be sufficiently representative of industry diversity, and personal opinions and experiences may influence results; differences in understanding of terminology in questionnaires may lead to response bias, and external factors such as time, resources, or emotional state may interfere with the assessment process. In addition, there are limitations to the applicability of the requirements boilerplate across different industries or projects, and statistical analysis methods may also have an impact on the results. Improvements such as increasing sample diversity, using double-blind assessments, clarifying terminology, and conducting more extensive empirical testing could be implemented to increase the validity of the study.

7. CONCLUSION

In summary, this research addresses the problems of ambiguity and unclear structure in Chinese software requirement expression, which can negatively affect the understanding and

implementation of software projects. By developing a set of Chinese-based software requirement boilerplate, this research provides a practical solution aimed at improving the clarity, completeness, singularity, verifiability, consistency ,and comprehensibility of requirement expression. The boilerplate, designed based on Chinese grammar specifications and international standards to ensure their applicability to various software industries. Through evaluation by linguists evaluation and feedback from industry professionals, we have validated the effectiveness of the template and demonstrated its great potential to improve the quality and efficiency of software requirements.

Looking ahead, there are many aspects of our work that we would like to explore in greater depth. First, we will continue to collect and analyse feedback from different software industries and application domains to further optimize and refine the templates so that they are more closely aligned with the habits of requirements expression in the actual development process. Second, we plan to introduce artificial intelligence and natural language processing technologies into the application of the templates to automate the checking and correction of requirement descriptions, and further improve the efficiency and accuracy of requirement document generation. Finally, we are looking forward to cooperating with more international counterparts to explore the standardization path of cross-lingual and cross-cultural software requirement expression and contribute to the collaborative development of the global software industry.

REFERENCES

[1] Y. U. Pabuccu, I. Yel, A. B. Helvacioglu, and B. N. Asa, "The requirement cube: A requirement template for business, user, and functional requirements with 5W1H approach," International Journal of Information System Modeling and Design, vol. 13, no. 1, pp. 1–18, 2022, doi: 10.4018/IJISMD.297046.

[2] C. Antoniou, K. Kravari, and N. Bassiliades, "Semantic requirements construction using ontologies and boilerplates," Data & Knowledge Engineering, vol. 152, p. 102323, 2024, doi: 10.1016/j.datak.2024.102323.

[3] J. W. Lim, T. K. Chiew, M. T. Su, S. Ong, H. Subramaniam, M. B. Mustafa, and Y. K. Chiam, "Test case information extraction from requirements specifications using NLP-based unified boilerplate approach," Journal of Systems and Software, vol. 211, p. 112005, 2024, doi: 10.1016/j.jss.2024.112005.

[4] Y. J. Zhao, Ambiguity Phenomena and Differentiation Methods in Modern Chinese, Encyclopedic Knowledge, no. 12, pp. 73–74, 2023.

[5] G. C. Oztekin and G. G. Menekse Dalveren, "Structured SRS for e-Government services with boilerplate design and interface," IEEE Access, vol. 11, pp. 62906–62917, 2023, doi: 10.1109/ACCESS.2023.3287882.

[6] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (EARS)," in Proc. 2009 17th IEEE Int. Requirements Engineering Conf., 2009, pp. 317–322, doi: 10.1109/RE.2009.9.

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

[8] S. Khalid, U. Rasheed, M. Muneer, W. H. Butt, R. Mehmood, and U. Qamar, "Common problems in software requirement engineering process: An overview of Pakistani software industry," in Proc. 2023 4th Int. Conf. Advancements in Computational Sciences (ICACS), 2023, pp. 1–6, doi: 10.1109/ICACS55311.2023.10089703.

[9] F. Huang, "Software requirement criteria based on human errors," in Proc. 2021 IEEE Int. Symp. Software Reliability Engineering Workshops (ISSREW), 2021, pp. 77–82, doi: 10.1109/ISSREW53611.2021.00047.

[10] J. Tian, J. Yin, and L. Xiao, "Software requirements engineer's ability assessment method based on empirical software engineering," Wireless Communications and Mobile Computing, vol. 2022, p. e3617140, 2022, doi: 10.1155/2022/3617140.

[11] K. Großer, M. Rukavitsyna, and J. Jürjens, "A comparative evaluation of requirement template systems," in Proc. 2023 IEEE 31st Int. Requirements Engineering Conf. (RE), 2023, pp. 41–52.

[12] M. R. R. Ramesh and Ch. S. Reddy, "Metrics for software requirements specification quality quantification," Computers & Electrical Engineering, vol. 96, p. 107445, 2021, doi: 10.1016/j.compeleceng.2021.107445.

[13] IEEE, "IEEE Recommended Practice for Software Requirements Specifications," IEEE Std 830-1998, pp. 1–40, Oct. 20, 1998, doi: 10.1109/IEEESTD.1998.88286.

[14] ISO/IEC/IEEE, International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering - Redline, ISO/IEC/IEEE 29148:2018(E) - Redline, pp. 1–209, Nov. 30, 2018.

[15] GB/T 9385-2008, "Software Engineering—Documentation Guidelines for Software Requirements Specifications," China National Standard, 2008.

[16] H. Y. Wang, Overview of Software Requirements Engineering Technologies, Computer Science, vol. S2, pp. 766–779, 2022.

[17] D. Majumdar, S. Sengupta, A. Kanjilal, and S. Bhattacharya, "Automated requirements modeling with Adv-EARS," International Journal of Information Technology Convergence and Services, vol. 1, pp. 57–67, 2011, doi: 10.5121/ijitcs.2011.1406.

[18] C. Rupp and R. Joppich, "Anforderungsschablonen ? Der MASTeR-Plan für gute Anforderungen," in Requirements-Engineering und -Management, pp. 215–246, Carl Hanser Verlag GmbH & Co. KG, 2014, doi: 10.3139/9783446443136.010.

[19] K. Thongglin, S. Cardey, and P. Greenfield, "Thai software requirements specification pattern," in Proc. 2013 IEEE 12th Int. Conf. Intelligent Software Methodologies, Tools and Techniques (SoMeT), Sept. 2013, pp. 179–184, doi: 10.1109/SoMeT.2013.6645650.

[20] R. D. G. Apaza, J. E. M. Barrios, D. A. I. Becerra, and J. A. H. Quispe, "ERS-TOOL: Hybrid model for software requirements elicitation in Spanish language," in Proc. Int. Conf. Geoinformatics and Data Analysis, 2018, pp. 27–30, doi: 10.1145/3220228.3220255.

[21] C. M. Keet and L. Khumalo, "Toward a knowledge-to-text controlled natural language of isiZulu," Language Resources and Evaluation, vol. 51, no. 1, pp. 131–157, 2017, doi: 10.1007/s10579-016-9340-0.

[22] IEEE, IEEE Std 830-1998 IEEE Standard for Software Requirements Specification, IEEE Computer Society, 1998.

[23] Y. M. Zang, Viewing Contextual Ambiguity from the Perspective of Chinese Language Teaching, Literary Education (Part II), no. 04, pp. 24–26, 2022, doi: 10.16692/j.cnki.wxjyx.2022.04.007.

[24] P. Yang, Analysis of Grammatical Ambiguity in Modern Chinese, Cranflower (Zhong), no. 09, pp. 119–121, 2023.

[25] G. Chen, Study of Ambiguity in Relational Markers and Relative Clauses, Journal of

Anhui University of Science and Technology, vol. 1, pp. 59–63, 2023.

[26] A. Fantechi, S. Gnesi, and L. Semini, "VIBE: Looking for Variability In amBiguous rEquirements," Journal of Systems and Software, vol. 195, p. 111540, 2023, doi: 10.1016/j.jss.2022.111540.

[27] Q. F. Cai, Analysis of Ambiguity Phenomena in Chinese, Cai Zhi, vol. 15, p. 213, 2019.

[28] Y. Y. Song, Application of Word Boundary Information in Chinese Segmentation Ambiguity, Chinese Character Culture, vol. 21, pp. 105–108, 2022.

[29] Z. Z. Zheng, A Study on the Differentiation of Ambiguous Structures in Modern Chinese Based on Quantification Theory, M.A. thesis, Hebei University, 2022.

[30] X. L. Shen, On the Chinese Nature of Chinese, Global Chinese Development Studies, no. 01, pp. 15–35, 2024.

[31] L. D. M. Barbosa, C. S. Cerqueira, and A. E. C. Da Cunha, "Natural language requirements boilerplates: An integrative literature review," Revista de Gestão e Secretariado (Management and Administrative Professional Review), vol. 14, no. 8, pp. 13444–13476, 2023, doi: 10.7769/gesec.v14i8.2610.

[32] Thangaratinam, S., & Redman, C. W. (2005). The delphi technique. The obstetrician & gynaecologist, 7(2), 120-125.

[33] D. Beiderbeck, N. Frevel, H. A. von der Gracht, S. L. Schmidt, and V. M. Schweitzer, "Preparing, conducting, and analyzing Delphi surveys: Cross-disciplinary practices, new directions, and advancements," MethodsX, vol. 8, p. 101401, 2021.

[34] H. Taherdoost, S. Sahibuddin, and N. Jalaliyoon, "Exploratory factor analysis; concepts and theory," Adv. Appl. Pure Math., vol. 27, pp. 375–382, Aug. 2022.

[35] M. Amirrudin, K. Nasution, and S. Supahar, "Effect of variability on Cronbach alpha reliability in research practice," Jurnal Matematika, Statistika dan Komputasi, vol. 17, no. 2, pp. 223–230, 2021.

[36] Oztekin, G. C., & Menekse Dalveren, G. G. (2023). Structured SRS for e-Government Services With Boilerplate Design and Interface. IEEE Access, 11, 62906–62917. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3287882

[37] Großer, K., Rukavitsyna, M., & Jürjens, J. (2023, September). A Comparative Evaluation of Requirement Template Systems. In 2023 IEEE 31st International Requirements Engineering Conference (RE) (pp. 41-52). IEEE.

[38] Antoniou, C., & Bassiliades, N. (2024). A tool for requirements engineering using ontologies and boilerplate. Automated Software Engineering, 31(1), 5.

[39] Großer, K., Rukavitsyna, M., & Jürjens, J. (2024). A Comparative Evaluation of Requirement Template Systems (Summary). In Software Engineering 2024 (SE 2024) (pp. 47-48). Gesellschaft für Informatik eV.

[40] de Mello Barbosa, L., Cerqueira, C. S., & da Cunha, A. E. C. (2023). Natural language requirements boilerplates: an integrative literature review. Revista de Gestão e Secretariado, 14(8), 13444-13476.

[41] Bao, Y., Yang, Z., Yang, Y., Xie, J., Zhou, Y., Yue, T., Huang, Z., & Guo, P. (2021). An Automated Approach to Generate SysML Models from Restricted Natural Language Requirements in Chinese. Jisuanji Yanjiu yu Fazhan/Computer Research and Development, 58(4), 706–730. Scopus. https://doi.org/10.7544/issn1000-1239.2021.20200757

[42] Chen, J., Hu, B., Diao, W., & Huang, Y. (2023). Automatic Generation of SysML Requirement Models Based on Chinese Natural Language Requirements. Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering, 242–248. https://doi.org/10.1145/3573428.3573470

[43] Wang, Z., Yu, Q., Wang, J., Hu, Z., & Wang, A. (2023). Grammar Correction for Multiple Errors in Chinese Based on Prompt Templates. Applied Sciences, 13(15), 8858. https://doi.org/10.3390/app13158858

[44] Xu, Q., & Wang, Z. (2022). A Data-Driven Model for Automated Chinese Word Segmentation and POS Tagging. Computational Intelligence and Neuroscience, 2022, 7622392. https://doi.org/10.1155/2022/7622392

[45] Chai, W., & Wang, J. (2022). A Chinese Named Entity Recognition Method Based on Fusion of Character and Word Features. 2022 IEEE 14th International Conference on Advanced Infocomm Technology (ICAIT), 308–313. https://doi.org/10.1109/ICAIT56197.2022.9862628

[46] Zhu, J., Huang, S., Shi, Y., Wu, K., & Wang, Y. (2022). A Method of K-Means Clustering Based on TF-IDF for Software Requirements Documents Written in Chinese Language. IEICE TRANSACTIONS on Information and Systems, E105-D(4), 736–754.