A Method of Verifying Time-Response Requirements

Yuma MATSUMOTO†*, Nonmember, Takayuki OMORI†(a), Hiroya ITOGA†∗(b), and Atsushi OHNISHI†(c), Members

SUMMARY In order to verify the correctness of functional requirements, we have been developing a verification method of the correctness of functional requirements specification using the Requirements Frame model. In this paper, we propose a verification method of non-functional requirements specification in terms of time-response requirements written with a natural language. We established a verification method by extending the Requirements Frame model. We have also developed a prototype system based on the method using Java. The extended Requirements Frame model and the verification method will be illustrated with examples.

key words: non-functional requirements, NFR, time-response requirements, requirements frame, verification of non-functional requirements

1. Introduction

Software requirements specification (SRS) should hold some characteristics in IEEE Std 830 [6]. For example, it claims that an SRS should be correct, unambiguous, consistent, and complete. We have developed a requirements frame and requirements language named X-JRDL based on the requirements frame to improve the characteristics of software functional requirements [10]. We could detect lack of indispensable cases and wrong noun types using X-JRDL, but since this language aims to specify functional requirements, we could not improve the characteristics of non-functional requirements (NFR).

In this paper, we propose an extended requirements frame and a verification method of NFR based on the extended requirements frame model.

In the next section, we illustrate the original requirements frame model and the requirements language. In Sect. 3, we describe an extended requirements frame model and a verification method of NFR specification, especially time-response requirements specification with examples. In Sect. 4, related works will be described. In Sect. 5, we conclude our research.

2. Requirements Frame Model

Consider requirements of a library system of book retrieval as below.

There exist users, cards of retrieval of books, and Identifier (ID) number of each book. Users are human-type objects. Cards and ID are data-type objects. Cards are classified into authors-cards that are sorted by author’s name in alphabetical order, and title-cards sorted by title. A user can retrieve books with these cards.

A requirements definer first identifies objects (nouns), object types (attributes) in a target system. Secondly he defines operations among objects (verbs) and roles of the operations (cases), and then constructs requirements sentences. The “cases” mean concept about agents, objects, goals of the operations [13]. Thus, a requirement sentence includes nouns and verbs as its components, and there exist roles of objects as relations among the components. A particular functional requirement may be defined with several sentences. Our requirements model named Requirements Frame Model has been developed to easily represent the above structures. It involves two kinds of frames; a frame of noun level and a frame of sentence level [10].

2.1 Noun Frame

The first frame is the Noun Frame, a frame whose components are nouns and their types. Table 1 shows the noun types provided to specify file-oriented software requirements. A new noun appearing in a requirements description will be classified into one of these types.

2.2 Case Frame

The second frame is the Case Frame, a frame whose com-

| Table 1 | Noun types of the noun frame. |
|---------|-----------------------------|
| Type of noun | Meaning |
| human | active and external object |
| function | active and internal object |
| file | passive object of information set |
| data | passive object of a single information |
| control | passive object for control transition |
| device | passive object of an instrument |
components are nouns, verbs and cases. We provided seven different cases: agent, goal, instrument, key, object, operation and source. We also provided 16 different concepts including data flow, control flow, data creation, file manipulation, data comparison, and structure of data/file/function. There are several verbs to represent one of these concepts. For example, to specify a concept data flow, we may use input, output, print out, display, send, and so on. A requirements definer can use any verbs as far as it can be categorized in these 16 concepts provided.

We prepared these concepts to specify requirements of a file-oriented business data processing domain. When a user wants to write requirements of another domain, he may need a verb not categorized into these concepts. In such a case, he can use a new verb if he defines its case structure. Since a newly defined verb, its concept, and its case structure can be registered in the verb dictionary, he can use his own verbs as well as provided verbs.

The 16 concepts (10 verb type concepts and 6 adjective type concepts) are shown in Table 2.

| Concept | Meaning            |
|---------|--------------------|
| DFLOW   | Data flow          |
| CFLOW   | Control flow       |
| ANDSUB  | And-tree structure |
| ORSUB   | Or-tree structure  |
| GEN     | Data creation      |
| RET     | Retrieve a record in a file |
| UPDATE  | Update a record in a file |
| DEL     | Delete a record in a file |
| INS     | Insert a record in a file |
| MANIP   | File manipulation  |
| EQ, NE, LT, GT, LE, GE | Comparison operators |

The Case Frame defines case structures of these concepts. For example, the data flow (DFLOW) concept has object, source, goal, and instrument cases. The object case corresponds to data which is transferred from the source case object to the goal case object. So, a noun assigned to the object case should be a data type object. A noun in the source or goal cases should be either a human or a function type object. If and only if a human type object is assigned to source or goal cases, some device type object should be specified as an instrument case. These are illustrated in Fig. 1. Each concept has its own case structure.

The Case Frame enables to detect illegal usage of data and lack of cases. Suppose a requirement sentence, “A user enters a retrieval command with a terminal.” Since the objective is “a retrieval command” that is data type noun, “enters” should be categorized into the DFLOW concept. With the Case Frame of the DFLOW, this sentence will be analyzed as shown in Table 3.

In this sentence the goal case object is omitted. The case structure of DFLOW says the goal case should be a noun of function type or human type. Previously specified nouns of the type become candidates of the omitted case. In this way, a requirement sentence is transformed into an internal representation named CRD (Conceptual Requirements Description). CRD is exactly based on the Noun Frame and the Case Frame.

2.3 Requirements Language: X-JRDL

We have developed a text-based requirements language named X-JRDL. This is based on the Requirements Frame model.

In X-JRDL, compound sentences and complex sentences will be divided into simple sentences each of which has just one verb for analysis with the Case Frame. These simple sentences are transformed into CRD. We adopted X-JRDL as a requirements language in the course of Information Systems, graduate school of Information Science, Kyoto University. Students specified SRSs of 50–500 sentences with this language. In this course we found 85% of sentences were interpretably accepted by the X-JRDL analyzer and others needed to be pre-edited.

2.4 Analysis of X-JRDL Description

An X-JRDL description is analyzed through three interpreters. Since X-JRDL allows compound sentences and complex sentences, a surface interpreter divides them into simple sentences. Another interpreter, word interpreter, fills a case structure consulting with dictionaries. Since a noun is interpreted with its type, the noun dictionary holds a name and its type. A verb (or an adjective) is interpreted with its corresponding concept. In the case of pronoun and omission of nouns, its type will be guessed with the Case Frame. A sentence interpreter transforms a simple sentence transformed into CRD with checking lacks of indispensable cases.
X-JRDL allows using pronouns and omission of nouns. We frequently come across such features in Japanese sentences. The X-JRDL analyzer automatically assigns a concrete word into a pronoun or a lacked case.

Conjunctions are used to write down compound sentences and complex sentences. The analyzer divides such a sentence into a set of simple sentences. The X-JRDL analyzer has a dictionary of nouns, verbs and adjectives. When a requirements definer uses a word which is not appeared in the dictionary, the analyzer guesses a type of new noun and a concept of new verb and adjective with the Requirements Frame. Table 4 shows registered verbs. Really these verbs are Japanese verbs. The analyzer can treat inflection of these verbs.

A same requirement can be described differently. For example, the previous requirement sentence “A user enters a retrieval command with a terminal” can be expressed as “A terminal receives a retrieval command from a user” or “A retrieval command can be passed from a user to a terminal.” The CRDs of these three sentences are exactly the same. In other words, if the CRDs are the same, the meanings of requirement sentences are the same.

Since X-JRDL is focused to express functional requirements, NFR cannot be described. Because it is very important to improve the quality of NFR [2], we would like to propose a verification method of NFR, in terms of time-response requirements.

In case of analysis of functional requirements based on the case grammar, each noun corresponds to a case as shown in Table 4. In contrast, NFR can be expressed with adjectives, adjective verbs, or adverbs. So, it is difficult to analyze NFR based on the case grammar, since adjectives, adjective verbs, adverbs cannot be clearly corresponded to cases. In this paper, we focus on time-response requirements, because time-response requirements include expressions of time-response. We can analyze expressions of time-response by detecting numerals and/or units.

### Table 4 Verbs in the dictionary.

| Concept     | part of registered verbs |
|-------------|--------------------------|
| DFLOW, CFLOW | pass, move, receive, input |
| ANDSUB, ORSUB | subpart, part, construct |
| GEN         | generate, produce, make |
| RET         | retrieve                 |
| INS         | insert, add              |
| UPDATE      | update                   |
| DEL         | delete                   |

### Table 5 Keywords related to time-response requirements.

| Keywords related to time-response requirements |
|-----------------------------------------------|
| response, respond, turnaround, second         |

3. Verification of Time-Response Requirements

By extending the Requirements Frame, we can analyze sentences of NFR and transformed into CRDs. As previously described, we focus on time-response requirements. This means our approach is limited to verification of time-response requirements. Time-response requirement is a requirement about interval between a user-command and the receipt of an action, result, or feedback from the system.

Saito et al. propose a machine learning approach to evaluate the clarity of NFR described in a Request For Proposal (RFP) written in a natural language [11], [12]. In this method, keywords related to NFR are extracted from a RFP, and mapped to each NFR category. Then, the clarity of NFR is modeled by the random forest with weight factors based on appearance frequency and context vectors. As a result of an experiment to evaluate the clarity (low, mid or high) of many NFR categories in 70 RFPs, the proposed method showed 69.8% match to the expert’s decision. They clarified 33 keywords for performance requirements including time-response requirements, network efficiency requirements, and memory resource requirements [12]. By omitting keywords for network efficiency and memory resource, we get 21 keywords. Some keywords in the 21 keywords include the same substring, such as “respond,” “response,” and “second.” We selected 4 keywords in Table 5, because these 4 keywords covers the 21 keywords.

With these 4 keywords, we retrieved requirements sentences in 20 RFPs. The results are shown in Fig. 2. In this figure, the same sentences are merged into one.

The seventh sentence is not a time-response requirement, because the agent (the presenter) is not a system or a function that responds. We do not regard a sentence whose responder is a human as a time-response requirement. Software developers who understand time-response requirements checked whether there are any other time-response requirements, but such requirements could not be found.

3.1 Requirements Frame for Time-Response Requirements

We introduced a new requirements frame to represent time-
Table 6  Transformed time-response requirements in Fig. 2.

| sentence# | agent   | goal                  | condition                  |
|------------|---------|-----------------------|----------------------------|
| 1          | -       | within three seconds  | normal time                |
| 1          | -       | within five seconds   | peak time                  |
| 2          | -       | within 3 seconds      | standard                   |
| 3          | -       | same or less than the response time of the as-is system | - |
| 4          | the server | within a reasonable time | - |
| 5          | -       | in 200 milliseconds   | bank card insertion        |
| 6          | -       | one second            | dial tone generation once a caller’s phone is detected |

Table 7  Transformed supposed time-response requirements.

| sentence_id | agent   | goal                  | condition |
|-------------|---------|-----------------------|-----------|
| a           | System A | within five seconds  | -         |
| b           | System A | within five seconds  | -         |
| c           | System A | within three seconds | -         |
| d           | System B | quickly               | -         |

Suppose four time-response requirements below.

a) The response time of system A shall be within five seconds.
b) System A shall respond within five seconds.
c) The response time of system A shall be within three seconds.
d) System B shall respond quickly.

These requirements can be transformed into internal representation shown in Table 7 respectively.

Surface representations of requirements a and b are different each other, but their meanings are same. We consider that these requirements are redundant. We can detect redundant requirements if there exist multiple requirements whose agent cases, whose goal cases, and whose condition cases are same nouns, respectively.

Requirement d is ambiguous, because this requirement is qualitatively described. We can detect ambiguous requirements in terms of time-response if there exists a requirement whose goal case is missing or qualitative.

Requirements a and c are inconsistent, because the values of time-response for the same system are different each other. In this paper, the consistency of time-response requirement means that the time-response requirement is free of conflicts with other time-response requirements. We can detect inconsistent requirements, if there exist two or more
1. Retrieve time-response requirements in an SRS using the four keywords.
2. User checks whether each of retrieved requirements is time-response requirements or not, and omits non-time-response requirements.
3. The time-response requirements will be analyzed with a natural language processor in order to analyze the syntax of the sentences.
4. Transform the analyzed results by the natural language processor into internal representation based on the extended requirements frame model.
5. Detect the redundancy, inconsistency, incompleteness, or ambiguity of the time-response requirements.

**Fig. 4** Verification Procedure of time-response requirements.

![Diagram](image)

**Fig. 5** The whole system of verification of time-response requirements.

Verification procedures of time-response requirements are shown in Fig. 4. In the first step, we use the 4 keywords to retrieve time-response requirements from requirements documents. However, not time-response requirements may be retrieved, so software developer judges whether time-response requirements are correctly retrieved or not in the second step. In the third step, time-response requirements are preprocessed. This step is still under development. In the fourth step, preprocessed time-response requirements are transformed into internal representation. Lastly, the four qualities of the time-response requirements described in Sect. 3.2 are checked.

In Table 6, the first three requirements have no agent cases. The first two requirements are not inconsistent, because the condition cases are different, although the missing agent cases are same and goal cases are different. The third requirement in Fig. 2 is ambiguous if the response time of the as-is system cannot be referred. The fourth requirement in Fig. 2 is ambiguous, too, because the goal case is qualitative and not quantitative.

We could detect the ambiguity, the inconsistency, the completeness, and the redundancy of time-response requirements with our method. Actually, the redundancy of requirements is not an error, but in case of modification if one requirement is changed and the other is not changed, this modification may cause the inconsistency.

**3.3 Prototype**

Figure 5 shows the whole system of verification of time-
response requirements. In Fig. 5, flowchart symbols are used. The numbers in this figure correspond to step numbers in Fig. 4, respectively. The step number, 2 is missing in Fig. 5, because this step will not be implemented.

We adopted an existing text-based retrieval system for retrieval with the four keywords. Retrieved time-response requirements are automatically processed by a natural language processor or manually processed by a user, and then transformed into internal representation based on the extended Requirements Frame. The error detecting system gets the internal representation and produces a report of detected errors.

We have developed a prototype system of error detection using Java with Eclipse 4.4 Luna. The number of source code lines is about 500. This is a 2 man-month product. This system supports the first, the fourth and the fifth steps of the verification procedure in Fig. 4.

Figures 6, 7, and 8 show snapshots of detecting an error with the prototype system. Original messages of the prototype system are in Japanese, so we added messages in English for English readers.

Using the prototype system, we could detect two ambiguous time-response requirements in 20 RFPs. These are the 3rd sentence and the 4th sentence in Fig. 2. The 4th sentence is really ambiguous, because the time-response requirement is defined qualitatively. In contrast, the 3rd sentence may not be ambiguous, if the response time of as-is system is quantitatively specified in other SRS. This requirement may be ambiguous, if the response time of as-is system is qualitatively specified in other SRS or not clearly specified. Inconsistent requirements and redundant requirements are not included in the 20 RFPs, but we added such errors by hand and detected them shown in Figs. 6 and 7.

3.4 Discussion

Our method enables to detect the redundancy, ambiguity, inconsistency, and incompleteness of time-response requirements. We supposed that a time-response requirement is expressed as a single sentence and we could analyze a time-response requirement sentence using the extended Requirements Frame described in Sect. 3.1. Because this frame is independent of requirements language, our verification method can be applied to SRSs with Japanese and SRSs with English. However, a time-response requirement may be expressed as multiple sentences or expressed as a sentence and a table like the second sentence in Fig. 9.

In Fig. 9, each time-response requirement is expressed as non-single sentence. In such a case, it is difficult to analyze a time-response requirement with the extended Requirements Frame model. However such requirements can be retrieved with the keywords shown in Table 5. By transforming such requirements into a single sentence by hand, we could apply our method to the transformed sentence.

As for the ambiguity, if condition case is missing, or condition case is specified qualitatively, we regard that the time-response requirement is ambiguous. There exist other causes producing the ambiguity, such as omission of nouns, use of pronouns, and grammatical ambiguities. The ambiguity of these causes can be detected by hand at the 2nd step of Fig. 2. Another cause is words with multiple meanings. The ambiguity of this cause can be detected with a dictionary or a thesaurus.

As for the completeness, users have to specify systems
that time-response requirements should be described. By checking whether time-response requirements whose agents are included in the specified systems or not, the completeness can be checked.

4. Related Works

The NFR-frame work is a goal-oriented analysis method for non-functional aspects of a target system [2]. Our method aims to verify the characteristics of an SRS, instead.

In [3], performance requirements written with a structured language will be transformed into Petri-net model and then the inconsistency and ambiguity will be checked. Our method enables to detect not only the inconsistency and ambiguity, but also the incompleteness and redundancy in requirements written with a natural language.

In [4], [5], use-case oriented verification method of NFR is proposed. This method is useful to check scenarios or use-case descriptions, but is not suitable to check SRSs.

In [9], rule-based checking method of NFR is proposed. This method checks whether NFR are specified or not and do not check the characteristics of NFR.

In [1], a formal verification method of NFR is proposed, but it is not suitable for SRSs in a natural language.

In [7], the importance of quantitative requirements is claimed, but they do not verify the characteristics of NFR.

In [11], [12], keywords for NFR including time-response requirements are proposed, but they do not check the qualities of NFR.

5. Conclusion

In this paper, we proposed a verification method of the unambiguity, the consistency, the completeness, and the redundancy of time-response requirements in SRS written with a natural language. We could detect ambiguous, inconsistent, incomplete, and/or redundant time-response requirements with our method. We also developed a prototype system based on the method.

Evaluation of the method by applying to other SRSs, and the establishment of verification method of other NFR are left as future works.

Acknowledgements

We thank to under graduate students Shouta Kasai, Sayaka Shirai, Satsuki Yamada, and other members of Software Engineering laboratory, Department of Computer Science, Ritsumeikan University for their contributions to the research. This research is partly supported by Grant-in-Aid for Scientific Research, Japan Society for the Promotion of Science, No.16K00112.

References

[1] B. Wei, B. Yin, Z. Jin, and D. Zowghi, “ξΣ: Automated reasoning tool for non-functional requirement goal models,” Proc. 19th International Requirements Engineering Conference (RE2011), pp.337–338, 2011.

[2] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering, Springer, 1999.

[3] M.D. Cin, “Structured language for specifications of quantitative requirements,” Proc. 5th IEEE International Symposium on High Assurance Systems Engineering (HASE), pp.221–227, 2000.

[4] L.M. Cysneiros and J.C.S. do Prado Leite, “Nonfunctional requirements: From elicitation to conceptual model,” IEEE Trans. Softw. Eng., vol.30, no.5, pp.328–350, 2005.

[5] A. Fatwanto and C. Boughton, “Analysis, specification and modeling of non-functional requirements for translatable model-driven development,” Proc. International Conference on Computational Intelligence and Security (CIS ‘08), pp.405–410, 2008.

[6] IEEE Standards Board: Section 4.3 “Characteristics of a good SRS,” IEEE Recommended Practice for Software Requirements Specifications,” IEEE830-1998, 1998.

[7] M. Irfan and H. Zhu, “Key role of value-oriented requirements to develop real-time database systems,” Proc. IEEE 2nd International Conference on Computing, Control and Industrial Engineering (CCIE), pp.405–408, 2011.

[8] ISO/IEC/IEEE: Systems and software engineering - Life cycle processes - Requirements engineering, International Standard, first edition, pp.10–11, 2011.

[9] H. Kaiya and A. Ohnishi, “Quality requirements analysis using requirements frames,” Proc. 11th International Conference on Quality Software (QSIC 2011), pp.198–207, 2011.

[10] A. Ohnishi, “Software requirements specification database based on requirements frame model,” Proc. 2nd International Conference on Requirements Engineering (ICRE ’96), pp.221–228, 1996.

[11] Y. Saito, A. Monden, and K. Matsumoto, “Evaluation of RFPs based on machine learning,” IPSJ SIG Technical Report, vol.2013-SE-179, no.5, pp.1–7, 2013 (in Japanese).

[12] Y. Saito, “Quantitative evaluation of non functional requirements in an early stage of software development,” Doctoral Dissertation, NAIST, NAIST-IS-DD1061202, Feb. 2015 (in Japanese).

[13] R. Shank, “Representation and understanding of text,” Machine Intelligence, vol.8, pp.575–607, Ellis Horwood Ltd., Cambridge, 1977.

Yuma Matsumoto received B. of Engineering degree from Ritsumeikan University in 2014. Currently he is with Panasonic System Design Co., LTD.
Takayuki Omori is an Associate Professor at Department of Computer Science, Ritsumeikan University. He obtained his Ph.D. in Engineering from Ritsumeikan University in 2008. From September 2013 to March 2014, he was a visiting assistant professor at the University of British Columbia. His current research interests include software evolution, and fine-grained code changes in software development.

Hiroya Itoga received M. and Dr. of Engineering degrees from University of Tsukuba in 1997 and 2002, respectively. He was a Research Associate, an Assistant Professor from 2003 to 2007, and an Associate Professor at Department of Computer Science, Ritsumeikan University from 2003 to 2018. Since 2018 he has been a Technical Staff at RIKEN Center for Biosystems Dynamics Research. Dr. Itoga is a member of IEEE Computer Society, ACM, Information Processing Society of Japan (IPSJ), and Japan Society for Software Science and Technology (JSSST).

Atsushi Ohnishi received B. of Engineering, M. of Engineering, and Dr. of Engineering degrees from Kyoto University in 1979, 1981, and 1988, respectively. He was a Research Associate and an Associate Professor of Kyoto University from 1983 to 1994. Since 1994 he has been a Professor at Department of Computer Science, Ritsumeikan University. He was a visiting scientist at University of California, Santa Barbara (UCSB) from March 1990 to February 1991 and also a visiting scientist at Georgia Institute of Technology in 2000. His current research interests include requirements engineering, object oriented analysis, and software design techniques. Dr. Ohnishi is a member of IEEE Computer Society, ACM, Information Processing Society of Japan (IPSJ), and Japan Society for Software Science and Technology (JSSST).