Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 159 (2019) 860–869

Procedia Computer Science

www.elsevier.com/locate/procedia

23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Verification method of reliability requirements

Satsuki Yamada[a], Takayuki Omori[a], Atsushi Ohnishi[a,*]

[a]*Department of Computer Science, Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan*

Abstract

It is very important to certify the accuracy of non-functional requirements in software development. In this paper, we focus on reliability requirements. We can correctly obtain reliability requirements from software requirements documents by using keywords related to reliability requirements. Retrieved requirements will be checked using a requirements frame model in order to verify the consistency, non-redundancy, unambiguity and completeness of these requirements. Our method will be illustrated with examples.

Keywords: derivation of non-functional requirements; verification of non-functional requirements; derivation of reliability requirements; verification of reliability requirements

1. Introduction

Software requirements specification (SRS) should hold some characteristics in IEEE Standards [6, 8]. 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 [12]. 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 reliable software requirements based on the extended requirements frame model.

In the next section, we illustrate the original requirements frame model and the requirements language. In Section 3, we introduce an extended requirements frame model for the verification. We describe a verification method of NFR specification, especially reliable requirements specification with examples in Section 4. In Section 5, we will discuss the proposed method in terms of the correctness of retrieval of reliability requirements. In Section 6, related works will be described. In Section 7, we conclude our research.

* Corresponding author. Tel.: +81-77-561-2693 ; fax: +81-77-561-5203.
E-mail address: ohnishi@cs.ritsumei.ac.jp

1877-0509
10.1016/j.procs.2019.09.245

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 an information set
data	passive object of a single information
control	passive object for a control transition
device	passive object of an instrument

2. Requirements Frame Model

Consider requirements of book retrieval of a library system 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 [15].

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 [12].

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

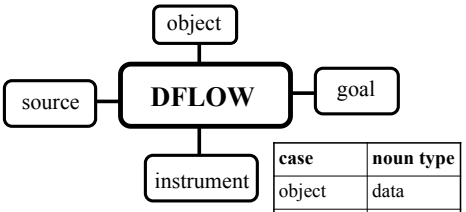
The second frame is the **Case Frame**, a frame whose components are nouns, verbs and cases. We provided seven different cases; *agent, goal, instrument, key, object, operation* and *source*. We also provided 16 different concepts including *data flow, control flow, data creation, file manipulation, 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.

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

Table 2. Concept of the Case Frame

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



case	noun type
object	data
source	function, human
goal	function, human
instrument	device

Fig. 1. Case Frame of the Concept, "DFLOW"

Table 3. Analysis of a requirement sentence

"A user enters a retrieval command with a terminal."

Concept	DFLOW
object	a retrieval command
source	a user
goal	** undefined **
instrument	a terminal

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 usages 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.

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

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, fulfills 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 sentence. 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 [1], we would like to propose a verification method of NFR, in terms of reliability 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 reliability requirements, because reliability requirements include expressions of reliability. We can analyze expressions of reliability by detecting numerals and/or units.

consideration, data, development, environment, failure, generation, guarantee, halt, maintenance, necessary, network, possibility, protection, provide, range, recovery, reliability, requirement, service, sufficient, system

Fig. 2. Word set A: Frequently used words in reliability requirements.

data, development, environment, generation, maintenance, necessary, network, possibility, protection, requirement, service, system

Fig. 3. Word set (A ∩ B): Frequently used words in both reliability and non-reliability requirements.

consideration, guarantee, provide, range, suffcient

Fig. 4. General terms (not included in IT terminology dictionary.)

3. Retrieval of reliability requirements

By extending the Requirements Frame, we can analyze sentences of NFR and transformed into CRDs. As previously described, we focus on reliability requirements. This means our approach is limited to verification of reliability requirements. Reliability requirement is a requirement to establish the required reliability of the software system at time of delivery.

3.1. Keywords of reliability requirements

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 [14, 13]. 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.

We propose a definition method of keywords of reliability requirements based on Saito's method as follows.

1. Manually extract reliability requirements sentences from requirements documents and also extract non-reliability requirements sentences from the documents.
2. Select words frequently used in the reliability requirements sentences as "word set A."
3. Select words frequently used in the non-reliability requirements sentences as "word set B."
4. Define "word set C" as $(A - (A \cap B) - \textit{general terms})$ where "general terms" mean words not included in IT terminology dictionary.
5. Define "word set D" as reliability terms in Japanese Industry standard(JIS) Z8115 [9]. JIS Z8115 is a Japanese translated version of IEC60050(191) [5].
6. Define "word set D*" as reduced reliability terms by extracting substrings from "word set D."
7. Define "word set E" as $(C \cup D*)$.

We adopted four requirements documents provided by Japanese government and open in the Internet and manually selected 30 reliability requirements sentences. Figure 2 shows frequently used words in the reliability requirements.

Figure 3 shows frequently and commonly used words in the reliability and non-reliability requirements. Figure 4 shows generally used words and not included in an IT dictionary. So, the word set C becomes as shown in Figure 5.

Figure 6 shows terms used in reliability in JIS Z8115. In Figure 6, some terms contain a common word, for example "reliability" is included in "operational reliability." In such a case, we adopt "reliability" as a keyword. We reduced Figure 6 simpler as shown in Figure 7. By merging Figure 7 with Figure 5 we got keywords as shown in Figure 8.

failure, halt, recovery, reliability

Fig. 5. Word set C as ($A - (A \cap B) - general\ terms$): Selected keywords derived from requirements documents.

capability, conformity, dependability, failure intensity, failure mode, failure ratio, inherent reliability, mean failure intensity, mean time between failure, MTBF, MTTF, MTTFF, operational reliability, reliability, reliability characteristics

Fig. 6. Word set D: Terms used in reliability in JIS Z8115.

capability, conformity, dependability, failure, MTBF, MTTF, MTTFF, reliability

Fig. 7. Word set D*: Reduced terms used in reliability in JIS Z8115.

capability, conformity, dependability, failure, halt, MTBF, MTTF, MTTFF, recovery, reliability

Fig. 8. Word set E as ($C \cup D*$): Adopted keywords of reliability.

Table 5. Retrieval result of reliability requirements.

	docA	docB	docC
reliability requirements	3	2	3
retrieved requirements	4	5	10
correctly retrieved requirements	3	2	3
Precision	0.75	0.4	0.3
Recall	1	1	1

3.2. Retrieval of reliability requirements using keywords

We adopted three requirements documents different from the four requirements documents used in selection of keywords. First we manually selected reliability requirements in the three documents. We assumed these requirements are correctly selected. Next, we retrieved requirements including keywords in Table 8. Table 5 shows retrieval result and both precision and recall ratio.

We could retrieve all of the reliability requirements, but retrieved some non-reliability requirements. Correctly retrieved reliability requirements are shown in Table 6.

In Table 5 the total number of reliability requirements is 8, but there exist a similar requirement and identical ones in them. The 8 requirements are reduced into 6 requirements as shown in Table 6.

3.3. Extended Requirements Frame for reliability requirements

We introduced a new and extended requirements frame to represent reliability requirements. We provided three cases of one action, that is, “HALT.” The three cases are “agent case,” “goal case,” and “condition case.” We provided

Table 6. Correctly retrieved reliability requirements.

id	**requirement**
a	System halt by failure should be once or less per year.
b	System failure by customer should be recovered within 3 hours.
c	In recovery, data will be restored by backup data.
d	In case of unauthorized alternation of data, data can be restored by backup data.
e	In case of failure, operatinal softwares should be heighly recovered.
f	In case of failure by terrible disaster, system and data will be recovered.

Table 7. Case structure of HALT.

id	agent	goal	condition
a	system	once or less per year	by failure

Table 8. Case structure of RECOVER.

id	agent	goal	instrument	condition
b	-	within 3 hour	-	system failure by customer
c	data	-	backup data	recovery
d	data	-	backup data	unauthorized alternation
e	operational softwares	highly	-	failure
f	system and data	-	-	failure by terrible disaster

[Condition][Subject][Action][Object][Constraint]

EXAMPLE: When signal x received [Condition], the system [Subject] shall set [Action] the signal x received bit [Object] within 2 seconds [Constraint].

Fig. 9. Example of requirement syntax [8]

four cases of the action, that is, "RECOVER." The four cases are 'agent case," "goal case," "instrument case," and "condition case." The agent case corresponds to a noun that operates. The goal case corresponds to reliability objective. So, goal case should be quantitative attributes. The condition case corresponds to condition or environment in the operation. The instrument case corresponds to device or tool in the operation.

In Table 6 actions, "halt," "recover," and "restore" are specified. We think "recover" and "restore" are similar with each other, and regard their concept as "RECOVER." We define a case structure of "HALT" for verb "halt" and the requirement "a" can be analyzed with this case structure as shown in Table 7.

We define a case structure of "RECOVER" for verb "recover" and "restore." The requirements labeled b, c, d, e, and f in Table 6 can be analyzed with this case structure as shown in Table 8. In this structure, agent, goal, and condition cases are indispensable, while instrument case is optional. In this table, "-" means that a concrete noun is not assigned. A software developer manually transformed the six retrieved reliability requirements in Table 6 into internal representation as shown in Table 7 and 8. He well knows the extended requirements frame.

In [8], an example of requirement syntax is proposed as shown in Fig. 9. Requirements sentences based on the syntax in [8] can be transformed into internal representations based on the extended Requirements Frame in Table 7 and 8, because the agent case in the extended Requirements Frame corresponds to "Subject" in Fig. 9, and the goal case corresponds to "Constraint" in Fig. 9.

4. Verification of reliability requirements

We propose a verification method of (1) the non-redundancy, (2) the unambiguity, (3) the consistency, and (4) the completeness of reliability requirements.

We can detect redundant requirements if there exist multiple requirements whose agent cases, whose goal cases, and whose condition cases are same nouns, (and whose instrument cases are same if action is RECOVER) respectively. As for the requirements (c) and (d), agent cases and instrument cases are same nouns, respectively, but conditions are different from each other. So, these requirements are not redundant.

We can detect ambiguous requirements if there exists a requirement whose goal case is missing or qualitative. We can also detect ambiguous requirements if there exists a requirement whose agent case is missing. As for the requirements (c), (d), and (f), goal cases are missing. So, we can regard them as ambiguous requirements. As for the requirement (b), agent case is missing. So, we can detect that the requirement (b) is ambiguous. As for the requirement (e), goal case is qualitative. So, we can detect the requirement (e) is also ambiguous.

1. Retrieve reliability requirements in an SRS using the four keywords.
2. User checks whether each of retrieved requirements is reliability requirements or not, and omits non-reliability requirements.
3. The reliability 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. 10. Verification Procedures of time-response requirements.

We can detect inconsistent requirements, if there exist two or more requirements whose agent cases are same and whose condition cases are same (and whose instrument cases are same if action is RECOVER), but whose goal cases are different. In Table 8, there are no inconsistent requirements.

We can detect lack of reliability requirements if there are no requirements whose actions are HALT or RECOVER. We can detect lack of reliability requirements of a certain function/system if there are no requirement whose agent case is the function/system and whose action is HALT or RECOVER. We also detect incomplete reliability requirements if their indispensable cases are missing.

Verification procedures of reliability requirements are shown in Fig. 10.

As for the requirement (b), agent case is missing. As for the requirements (c), (d) and (f), goal cases are missing. So we can detect that these four requirements are incomplete.

We can detect the ambiguity, the inconsistency, the completeness, and the redundancy of reliability 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.

4.1. Prototype

Fig. 11 shows the whole system of verification of reliability requirements. We use an existing text-based retrieval system as the Retrieval system in this figure. Retrieved reliability requirements are automatically processed by a natural language processor or manually processed by a user, and then transformed into internal representaion based on the extended Requirements Frame. The error detecting system gets the internal representation and produce a report of detected errors. We are developing a prototype system of error detection with Java.

5. Discussion

We have focused on reliability requirements and adopted 15 reliability terms as the word set D from the section of reliability in the glossary of terms used in dependability [9]. This glossary has 59 terms in the section of failure, and 11 terms in the section of availability. The 57 terms in the section of failure include the term “failure” and the 2 terms include “flaw.” All of the 11 terms in the section of availability include “availability.” The terms “flaw” and “availability” may be added to word set D, however, because the recall values of retrieval of reliability requirements are 1.0, we think adopted keywords are sufficient for retrival of reliability requirements from three requirements documents. We do not insist that these keywords are sufficient. We have to improve the keywords by applying to other requirements documents and keep high recall ratio.

The precision and the recall ratios of retrieving reliability requirements depend on the keywords in Table 8. By adding other keywords, the recall ratio may be improved while the precision ratio becomes worse. Actually, guidelines to developers in a requirements document are wrongly retrieved as reliability requirements in our experiments. Such wrongly retrieved sentences can be reduced by checking the agent cases of the sentences. We will focus on both concept of action and its agent of each sentence and improve the precision and recall ratio.

Our method enables to detect the redundancy, ambiguity, inconsistency, and incompleteness of reliability requirements. We supposed that a reliability requirement is expressed as a single sentence and we could analyze a reliability

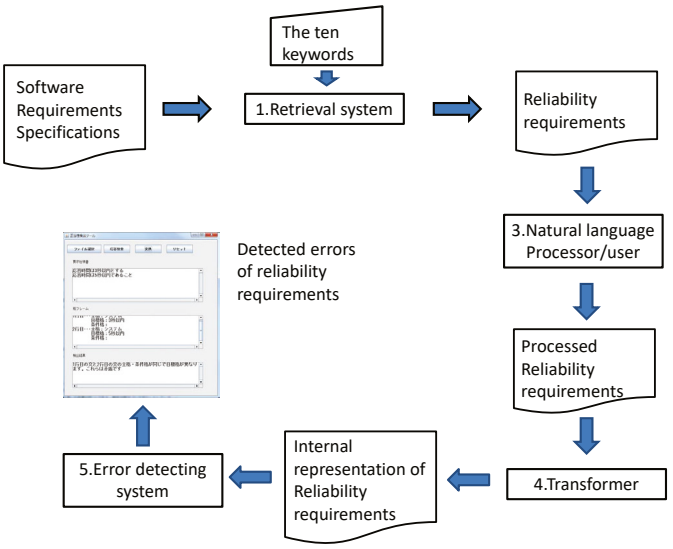


Fig. 11. Outline of verification system of reliability requirements.

1) The frequency of backup for data recovery are shown as follows. At peak-time it should be every one hour. At non-peak-time it should be every 3 hours.
2) The MTBF and MTTF of the system is shown in Table X.
3) The MTBF of the new system should be same as the MTBF of the current system.

Fig. 12. Reliability requirements with non-single sentence.

requirement sentence using the extended Requirements Frame described in Section 3.3. Because this frame is independent of requirements language, our verification method can be applied to SRSs with Japanese and SRSs with English.

However, a reliability requirement may be expressed as multiple sentences or expressed as a sentence and a table like the second sentence in Fig. 12. In Fig. 12, each reliability requirement is expressed as non-single sentence. In such a case, it is difficult to analyze them with the extended Requirements Frame model. However such requirements can be retrieved with the keywords shown in Table 8. By transforming such requirements into a single sentence by hand, we can apply our method to the transformed sentence.

As for the ambiguity, if a word specified in requirement sentence has multiple meanings, this sentence may be ambiguous. The ambiguity of this cause can be detected with a dictionary or a thesaurus.

6. Related Works

The NFR-frame work is a goal-oriented analysis method for non-functional aspects of a target system [1]. Our method aims to verify the characteristics of an SRS, instead.

In [2], 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 [3, 4], 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 [10], rule-based checking method of NFR is proposed. This method checks whether NFR are specified or not and does not check the characteristics of NFR. In [16], 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 [14], [13], keywords for NFR including reliability requirements are proposed, but they do not check the qualities of NFR. In [11], retrieval and verification methods of time-response requirements are proposed and a prototype based on the methods are implemented. These methods can be applied to time-response requirements, but cannot be applied to reliability requirements.

7. Conclusions

In this paper, we proposed a verification method of the unambiguity, the consistency, the completeness, and the redundancy of reliability requirements in SRS written with a natural language. We could detect ambiguous, inconsistent, incomplete, and/or redundant reliability requirements with our method.

We are now developing 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, Mr. Shouta Kasai, Ms. Sayaka Shirai, Mr. Takafumi Kinoshita, 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 and No. 19K11913.

References

[1] Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J., (1999). Non-Functional Requirements in Software Engineering. 1st ed., Springer US, New York.
[2] Cin, M.D., (2000). Structured language for specifications of quantitative requirements, in: HASE2000. 5th International Symposium on High Assurance Systems Engineering, IEEE Computer Society. pp. 221–227.
[3] Cysneiros, L.M., Sampaio, P., Leite, J.C.S.P., (2005). Non-functional requirements: from elicitation to conceptual model. Transaction on Software Engineering 30, 328–350.
[4] Fatwanto, A., Boughton, C., (2008). Analysis, specification and modeling of non-functional requirements for translative model-driven development, in: CIS'08. International Conference on Computational Intelligence and Security, pp. 405–410.
[5] IEC, (1990). Dependability-common terms. International Electrotechnical Vocabulary Chapter 191: Dependability and quality of service Part 1 .
[6] IEEE, (1998). Characteristics of a good srs. IEEE Recommended Practice for Software Requirements Specifications IEEE830-1998.
[7] Irfan, M., Hong, Z., (2011). Key role of value-oriented requirements to develop real-time database systems, in: CCIE2011. 2nd International Conference on Computing, Control and Industrial Engineering, IEEE Computer Society. pp. 405–408.
[8] ISO/IEC/IEEE, (2011). Systems and software engineering - life cycle processes- requirements engineering. International Standard , 10–11.
[9] JIS, (2000). Glossary of terms in dependability. JIS Z8115 .
[10] Kaiya, H., Ohnishi, A., (2011). Quality requirements analysis using requirements frames, in: QSIC 2011. 11th International Conference on Quality Software, IEEE Computer Society. pp. 198–207.
[11] Matsumoto, Y., Omori, T., Itoga, H., Ohnishi, A., (2018). A method of verifying time-response requirements. Transactions on Information and Systems E101-D, 1725–1732.
[12] Ohnishi, A., (1996). Software requirements specification database based on requirements frame model, in: ICRE '96. 2nd International Conference on Requirements Engineering, IEEE Computer Society. pp. 221–228.
[13] Saito, Y., (2015). Quantitative Evaluation of Non Functional Requirements in an Early Stage of Software Development (in Japanese). Naist-is-dd1061202 ed., Doctoral Dissertation, Nara Institute of Science and Technology (NAIST), Japan.
[14] Saito, Y., Monden, A., Matsumoto, K., (2013). Evaluation of rfps based on machine learning (in japanese). Special Interest Group Software Engineering, Technical Report 2013-SE-179, 1–7.
[15] Shank, R., (1977). Representation and Understanding of Text. volume 8. Ellis Horwood Ltd., Cambridge.
[16] Wei, B., Yin, B., Jin, Z., Zowghi, D., (2011). r ∑: Automated reasoning tool for non-functional requirement goal models, in: RE2011. 19th International Requirements Engineering Conference, IEEE Computer Society.