Regular Paper

Tool Supported Detection of Omissions by Comparing Words between Requirements and Design Document

Takehiro Wakabayashi[1] Shuji Morisaki[1] Norimitsu Kasai[2] Noritoshi Atsumi[3]
Shuichiro Yamamoto[1]

Received: May 28, 2019, Accepted: November 7, 2019

Abstract: This article proposes a tool supported approach to detect omitted requirements that are not implemented in a corresponding architectural design document using difference sets of words or word senses between a software requirements specification document and a software architectural design document. First, the proposed approach extracts sets of single-words, multi-words, and word senses that appear in a requirements specification document but do not appear in the corresponding design document using a natural language processing tool. Then, an architectural design document inspector validates whether each of the specified document with the single-words, multi-words, or word senses are implemented in the corresponding architectural design document using the sets as guides. Evaluation 1 investigated whether omitted requirements can be detected in design documents using the proposed approach. Evaluation 2 investigated the numbers of words that inspectors need to check for the proposed approach. The result of Evaluation 1 shows that omitted requirements are detected in all three pairs for real requirements specification documents and design documents. The result of Evaluation 2 shows that the numbers of words in the difference sets to those in the requirements specification documents vary from 18 to 83 % for the nine pairs of requirements specification documents and design documents.

Keywords: quality assurance, activity support (incl. communication support), verification/testing methods and environments, natural language analysis

1. Introduction

In embedded software development, a sequential software development process such as the waterfall model is widely used due to the many interfaces and external dependencies [1]. In sequential software development processes, software inspectors ensure that the descriptions in the software architectural design documents satisfy the requirements in the software requirements specification document. Inspectors validate that each requirement in the software requirements specification document is implemented and is specified in the design elements. Those inspections require a large amount of effort because they are conducted manually [2], [3].

Porter et.al classified the detected defects in inspection into omission, commission, and ambiguous types [4]. Investigating four software developments by using the GQM [5] framework, Mashiko and Basili demonstrated that an omission-type defect requires a larger effort to correct than a commission-type defect [6]. This is because, when correcting an omission type defect, a large amount of effort is required to add design elements without losing consistency with the existing design elements.

To increase the effectiveness and efficiency of inspections, tool supported approaches were proposed [7], [8], [9]. The approaches support code reviews and inspections by detecting candidates of source code defects and require pre-defined rules, templates, or testing code. Defining the specific rules, templates, and testing code to detect omitted requirements is more difficult than those to detect other types of defects such as inconsistency and incorrect defects because it may require the awareness of the potential omissions.

Toward a tool supported approach for detecting omitted requirements in architectural design documents, the authors investigated and identified words that met the following condition (1) and word senses that met the following condition (2) in five pairs of requirements specification documents and architectural design documents [10]:

(1) words that appear in the requirements specification document and do not appear in the corresponding architectural design document; and
(2) word sense of a homonymous word that appears in the requirements specification document and does not appear in the corresponding architectural design document.

However, the investigation in the article [10] is insufficient for real-world inspections. First, the architectural design documents in the investigation in the article were not intermediate versions that contained defects detected in inspections and testing but publicly available final versions that contained no defects detected in inspections and testing. Second, the number of words that the inspector needed to check was not investigated. Third, the ar-

[1] Nagoya University, Nagoya, Aichi 464–8601, Japan
[2] Communication Systems Center, Information Security Management and Investigation Department, Security Section, Mitsubishi Electric Co., Ltd., Amagasaki, Hyogo 661–8661, Japan
[3] Academic Center for Computing and Media Studies, Kyoto University, Kyoto 606–8501, Japan

ticle [10] does not consider multi-words, which help inspectors find and detect omissions more easily. For example, a multi-word "administration screen" helps inspectors identify the omitted description more than single-words "administration" and "screen".

This article empirically investigates whether such words can help an inspector detect omissions in an architectural design document and the number of words that an inspector needs to check. More specifically, the research questions are formulated as follows:

RQ.1 Can the inspector identify omitted requirements in an architectural design document by checking the difference sets between words in the requirements specification document and words in the corresponding architectural design document?

RQ.2 How many words in the difference set does the inspector need to check?

The article is organized as follows. In Section 2, we propose an approach to detect omissions in an architectural design document by extracting difference sets of words and word senses. Section 3 describes the evaluation settings to investigate whether inspectors can detect omissions using the proposed approach and the numbers of words in the difference sets. The results and discussion are presented in Sections 4 and 5, respectively. Section 6 introduces related work. Section 7 concludes the study.

2. Proposed Approach

2.1 Prerequisite

The proposed approach helps inspectors detect requirements that are not implemented in the corresponding architectural design document by showing the difference set of words between the requirements specification document and the architectural design document. The proposed approach assumes that the documents are written in natural language. The difference set is obtained by natural language processing to reduce the additional effort for the proposed approach. The proposed approach obtains three types of difference sets: difference sets of single-words, multi-words, and word senses.

As shown in **Fig. 1**, the proposed approach can be used in a self-check phase prior to the inspection, in the inspection phase, and in the final checkup phase after the inspection. In the self-check phase, authors of the architectural design document check for omissions in the parts they are responsible for prior to the inspection. In the inspection phase, the inspectors check for omissions using the proposed approach as one of their guides. In the final checkup phase, the inspectors of the independent inspectors check for omissions using the proposed approach after the inspection.

The proposed approach assumes that the authors, inspectors, and independent inspectors have knowledge of the target software and understand the structure of the documents so that they can locate and check potential omissions in the software architectural design document. If the inspector has a little knowledge of the documents, s/he will try to find by keyword search for the document.

The set of differences from the requirements $D(R, A)$ between the requirements specification document and the corresponding architectural design document is defined as:

$$D(R, A) = \{r \mid r \in S(R), r \notin I(A)\}$$

where r is the requirement, $S(R)$ is the set of requirements specified in the requirements specification document R, and $I(A)$ is the set of requirements implemented in the architectural design document A. The difference sets of words between the requirements specification document R and the corresponding architectural design document A are the difference sets of single-words Δ_s, multi-words Δ_m, and word senses Δ_h defined as follows.

$\Delta_s(R, A) = \{w_s | w_s \in W_s(R) \wedge w_s \notin W_s(A)\}$, where w_s is a single-word, $W_s(R)$ is the set of single-words that appear in the requirements specification document R, and $W_s(A)$ is the set of single-words that appear in the architectural design document A.

The following is a small example of R, A, and $\Delta_s(R, A)$. The omitted requirement is the transmission priority. Words "same", "time, "defined", and "priority" help inspectors find omitted requirement in the design document.

> R: When a transmission request from port α or β arrives, send data to appropriate network interfaces. If requests from α and β arrive at the same time, the program first sends α and then β as defined in the transmission priority.
> A: The monitoring program monitors I/O port α and β. If the program receives a transmission request from port α, the program sends the data to network interface X. If the program receives a transmission request from port β, the program sends the data to network interface Y.
> $\Delta_s(R, A)$ = {When, arrive, appropriate, at, same, time, first, then, as, defined, priority}
> $D(R, A)$ = {If requests from α and β arrive at the same time, the program first sends α and then β as defined in the transmission priority.}

$\Delta_m(R, A) = \{w_m | w_m \in W_m(R) \wedge w_m \notin W_m(A)\}$, where w_m is a multi-word, $W_m(R)$ is the set of multi-words that appear in the requirements specification document R, and $W_m(A)$ is the set of multi-words that appear in the architectural design document A. A multi-word consists of x single-words occurring within a maximum window size of y (with $y \geq x$) [11]. Multi-words can be more specific guides than single-words. For example, a multi-word "admin screen" is more specific than single-words "admin" and "screen".

$\Delta_h(R, A) = \{(w_h, m) | w_h \in W_s(R) \wedge w_h \in W_s(A), m \in M(R, w_h), M(R, w_h) \wedge \overline{M(A, w_h)} \neq \emptyset, |M(R, w_h)| \geq 2\}$, where (w_h, m) represents the word sense of a single-word w_h, $M(R, w_h)$ is the set of senses of word w_h that appears in the requirements specification document R, and $M(A, w_h)$ is the set of senses of word w_h that appears in the architectural design document A. For exam-

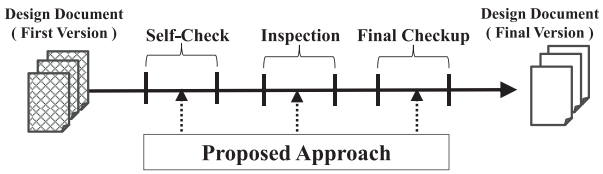


Fig. 1 Timings when the proposed approach can be used.

Table 1 Symbols list.

Symbol	Meaning
R	requirement specification document
A	architectural design document
A'	A which was the versions prior to the testing phase and was revised to correct the defects detected in the testing phase
$D(R, A)$	set of requirements in R, which not implemented in A
$\Delta_s(R, A)$	difference sets of single-words between R and A
$\Delta_m(R, A)$	difference sets of multi-words between R and A
$\Delta_h(R, A)$	difference sets of word senses between R and A
$W_s(R)$	sets of single-words in R
$W_m(R)$	sets of multi-words in R
$W_h(R)$	sets of word senses in R
$C_F(\Delta_s(R, A))$	single-words that are identified by the inspector as obviously irrelevant to omission in $\Delta_s(R, A)$
$C_F(\Delta_m(R, A))$	multi-words that are identified by the inspector as obviously irrelevant to omission in $\Delta_m(R, A)$
$C_F(\Delta_h(R, A))$	word senses that are identified by the inspector as obviously irrelevant to omission in $\Delta_h(R, A)$
$\Delta'_s(R, A)$	$\Delta_s(R, A)$ - $C_F(\Delta_s(R, A))$
$\Delta'_m(R, A)$	$\Delta_m(R, A)$ - $C_F(\Delta_m(R, A))$
$\Delta'_h(R, A)$	$\Delta_h(R, A)$ - $C_F(\Delta_h(R, A))$
$Ratio_s(R, A)$	ratio of $\Delta_s(R, A)$ to the number of $W_s(R)$
$Ratio_m(R, A)$	ratio of $\Delta_m(R, A)$ to the number of $W_m(R)$

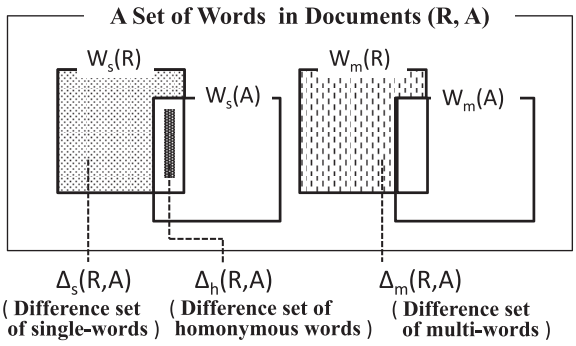
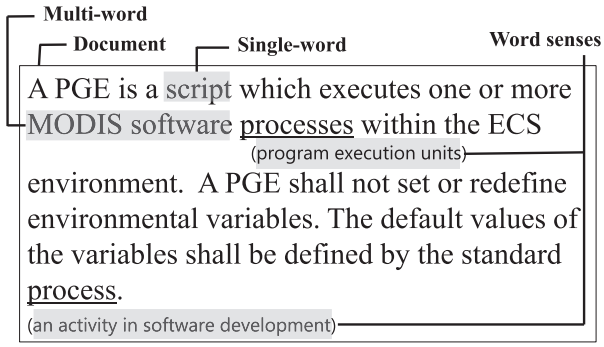


Fig. 2 $\Delta_s(R, A)$, $\Delta_m(R, A)$,and $\Delta_h(R, A)$.



Note that the word "process" has "program execution unit" and "an activity in software development".

Fig. 3 Example of a requirement specification document demonstrating single-word, multi-word, and word sense.

ple, the word "process" in requirements may have the meanings "program execution unit" and "an activity in software development". In the case that both of the meanings are included in the requirements, word senses can help detect omissions of either of the meanings.

Figure 2 shows a Venn diagram of Δ_s, Δ_m, and Δ_h. **Figure 3** shows an example of a document including a single-word, a multi-word, and a word sense.

Table 1 shows the symbols used in this paper.

2.2 Procedure

2.2.1 Step 1 Checking Omissions with Single-words

Step 1.1: This step extracts the sets of single-words $W_s(R)$ and $W_s(A)$ from the requirements specification document R and the corresponding architectural design document A using a morphological analyzer. The morphological analyzer automatically extracts words and filters stop words. Stop words such as *a*, *the*, and *that* have grammatical functions but do not carry any specific information [12]. After the filter, this step obtains the sets of words $W_s(R)$ and $W_s(A)$.

Step 1.2: This step obtains the difference set of the single-words $\Delta_s(R, A)$ from $W_s(R)$ and $W_s(A)$. Then, it obtains the set of possible omitted single-words $\Delta'_s(R, A)$ by excluding obviously irrelevant words $C_F(\Delta_s(R, A))$ from the difference set $\Delta_s(R, A)$. The inspector manually excludes the subset $C_F(\Delta_s(R, A))$ from the set of words $\Delta_s(R, A)$. The subset $C_F(\Delta_s(R, A))$ consists of words that are identified by the inspector as obviously irrelevant to omissions, such as the names of the development support tools and the hardware model number, without checking the architectural design document A. The set of possible omitted single-words $\Delta'_s(R, A)$ is defined as: $\Delta'_s(R, A) = \{\Delta_s(R, A) \wedge \overline{C_F(\Delta_s(R, A))}\}$. Finally, the inspector validates the architectural design document A with each single-word in the set of possible omitted single-words $\Delta'_s(R, A)$. Although the proposed approach does not assume a specific check procedure, the inspector can use visual checks as in software inspections and keyword search by selecting similar keywords to each of the single-words such as selecting "account" and "user ID" as search keywords for the single-word "identifier".

2.2.2 Step 2 Checking Omissions with Multi-words

Step 2.1: This step obtains the sets of multi-words $W_m(R)$ and $W_m(A)$ from the sets $W_s(R)$ and $W_s(A)$ obtained in Step 1.1. Algorithm 1 shows the procedure to obtain the multi-words. In **Fig. 4**, Algorithm 1 defines the multi-word's maximum length $mwl = 2$, and the multi-word's maximum span $mws = 1$ where mwl is the maximum number of combination of single-words and mws is the combination range of the single-words. Algorithm 1 is based on the algorithm proposed in the article [11].

Step 2.2: This step obtains the difference set of the multi-words $\Delta_m(R, A)$ from $W_m(R)$ and $W_m(A)$. Then, it obtains the set of pos-

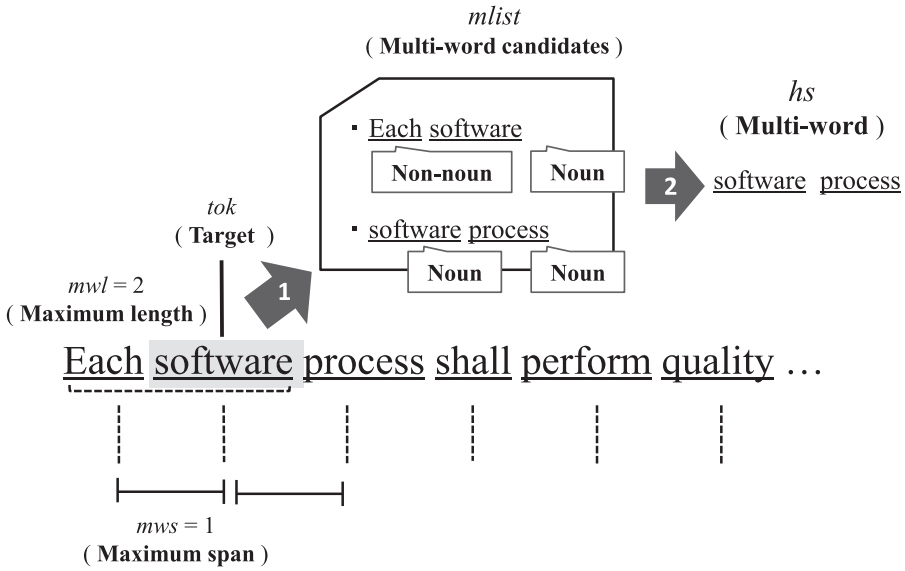


Fig. 4 Procedure for extracting multi-words.

Algorithm 1 Generating multi-word from single-words [*1].

Input:
ts, sequence of tokens in the document collection
mwl, maximum length of multi-word
mws, maximum span of words in multi-word
Output:
fs, a set of k features
Initialize:
hs := empty hashtable

```
for all tok in ts do
  Generate a list of multi-word tokens ending in tok.
  This list includes the single-word tok and uses the inputs mws and mwl.
  Call this list mlist.
  for all mtok in mlist do
    if hs contains mtok in hs then
      i := value of mtok in hs
      increment i by 1
    else
      i := 1
    end if
    store i as value of mtok in hs
  end for
end for
sk :=keys in hs sorted by decreasing value
output fs
```

sible omitted multi-words $\Delta'_m(R, A)$ by excluding obviously irrelevant multi-words from the difference set $\Delta_m(R, A)$. The inspector manually excludes the subset $C_F(\Delta_m(R, A))$ from the set of multi-words $\Delta_m(R, A)$, and obtains $\Delta'_m(R, A)$. Finally, the inspector validates A with each multi-word in the set of possible omitted multi-words $\Delta'_m(R, A)$ same as Step 1.2.

2.2.3 Step 3 Checking Omissions with Word Senses

Step 3.1: This step automatically obtains the set of candidate homonymous words $w_h(R, A)$ from the intersection set of single-words $W_s(R) \wedge W_s(A)$ using a dictionary. It refers to monolingual dictionary. When a glossary is attached to a document used in development and defines the meaning of a homonymous word, the glossary can be used as a dictionary. The set of homonymous words $w_h(R, A)$ is defined as: $w_h(R, A) = \{(w_h, m) | w_h \in W_s(R) \wedge w_h \in W_s(A), |M_d(w_h)| \geq 2\}$ from the requirements specification document R and the architectural design document A, where $M_d(w_h)$ is a set of word senses of word w_h defined in a dictionary d. Note that the homonymous words are extracted only by the definition of the dictionary. Thus, not all of the candidate words have two or more word senses in the documents.

Step 3.2: This step obtains the difference set of word senses $\Delta_h(R, A)$ from $w_h(R, A)$. The inspector identifies the word senses of a homonymous word $w_h(R, A)$ in R and A. The inspector manually excludes the subset $C_F(\Delta_h(R, A))$ from set of word senses $\Delta_h(R, A)$ and obtains $\Delta'_h(R, A)$. Finally, the inspector validates A with each word sense in the set of possible omitted word senses $\Delta'_h(R, A)$ same as Steps 1.2 and 2.2.

3. Evaluation

3.1 Overview

The evaluation investigates whether the inspector can detect omissions in architectural design documents using the proposed approach. This evaluation also investigates the number of words in the difference sets Δ_s and Δ_m. In Evaluation 1, the inspector was not involved in the software development but had experience in similar software developments. Word extractors were implemented using an open source morphological analyzer to automatically extract single-words, multi-words, and homonymous words from the documents. Two versions of word extractors with morphological analyzers were implemented for Japanese and English documents. Evaluation 1 obtains $\Delta'_s(R_x, A'_x)$, $\Delta'_m(R_x, A'_x)$, and $\Delta'_h(R_x, A'_x)$, which are the difference sets of the single-words, multi-words, and word senses between the intermediate version of the architectural design document A'_x and the corresponding requirements specification document R_x. Evaluation 1 extracts only multi-words consisting of two single-words ($mwl = 2$ and $mws = 1$). Because the number of extracted multi-words is

[*1] The algorithm is quoted from P.33 in the article [11] except filtering processing by word frequency.

likely enormous if the evaluation obtains multi-words consisting of three or more single-words, Evaluation 1 only obtains multi-words consisting of two single-words as a first step. In Evaluations 1.1, 1.2, and 1.3, an inspector manually validates whether the requirements specified with the single-words in $\Delta'_s(R_x, A'_x)$, multi-words in $\Delta'_m(R_x, A'_x)$, or word senses in $\Delta'_h(R_x, A'_x)$ are implemented in A'_x by visual checks as in software inspections and keyword search by selecting similar keywords. In Evaluation 1.4, the inspector investigates whether the single-words in $\Delta'_s(R_x, A'_x)$ and $C_F(\Delta_s(R_x, A'_x))$ are categorized into groups according to their characteristics. The inspector also investigates the multi-words in $\Delta'_m(R_x, A'_x)$ and $C_F(\Delta_m(R_x, A'_x))$ and the word senses in $\Delta'_h(R_x, A'_x)$ and $C_F(\Delta_h(R_x, A'_x))$.

Evaluation 2 counts the number of words in the difference sets of $\Delta'_s(R, A)$ and $\Delta'_m(R, A)$ for nine pairs of requirements specification documents and architectural design documents. Evaluation 2 also investigates correlations between the numbers of single-words and multi-words in the requirements specification documents and in the difference sets $\Delta'_s(R, A)$ and $\Delta'_m(R, A)$ for the nine pairs. Investigations on the difference set of the word senses $\Delta'_h(R, A)$ are not included in Evaluation 2 because accurate word senses in six pairs of documents could not be obtained by the authors.

3.2 Evaluation 1

3.2.1 Materials

The materials for Evaluation 1 consisted of three pairs of requirements specification documents and the corresponding architectural design documents of three different software systems for communication network monitoring systems x_1, x_2 and x_3. All of the systems were developed in a manufacturing company in Japan. The projects are waterfall-managed projects, which included systems integration and the development of controlling and monitoring software for various network devices. The requirements specification documents R_{x_1}, R_{x_2}, and R_{x_3} were final versions that were not revised after the requirement inspections. The corresponding architectural design documents A'_{x_1}, A'_{x_2}, and A'_{x_3} were the versions prior to the testing phase and were revised to correct the defects detected in the testing phase. The document A''_{x_1} was the version prior to the design inspection. The documents x_2 and x_3 prior to the design inspection were not available. One pair of the three pairs had one additional version of the architectural document that was the version prior to the design inspection. Three defect lists recorded in the testing phase and one defect list recorded in the design inspection were used as baseline and oracles to compare to the omissions detected by the proposed approach as the following reasons. In case studies, Kitchenham et al. suggest choosing standard practices in the organization as a baseline [13]. Some defects detected in the testing phase required corrections to the architectural design document. Each of the documents was developed by following the organization standard development process which has been refined in the organization over several decades. All three software systems have been in operation for five or more years in industry without needing architectural design defects to be corrected. The proposed approach was not used in any development activities including the design inspections. If the proposed approach detects omission defects, the inspector confirms whether the omission defects were included in the defect lists recorded in the design inspections and the testing phase after the evaluation. **Table 2** lists the sizes of the documents. All of the documents were written in Japanese. **Figure 5** shows the versions of the documents.

Table 2 Sizes of the documents R_x, A''_x, and A'_x.

Document	Number of pages	Number of single-words	Number of multi-words	Number of letters
R_{x_1}	186	1318	1889	87112
R_{x_2}	32	503	481	10724
R_{x_3}	56	663	974	26710
A''_{x_1}	134	887	875	29482
A'_{x_1}	134	1060	875	45064
A'_{x_2}	45	608	826	21468
A'_{x_3}	56	606	1036	25767

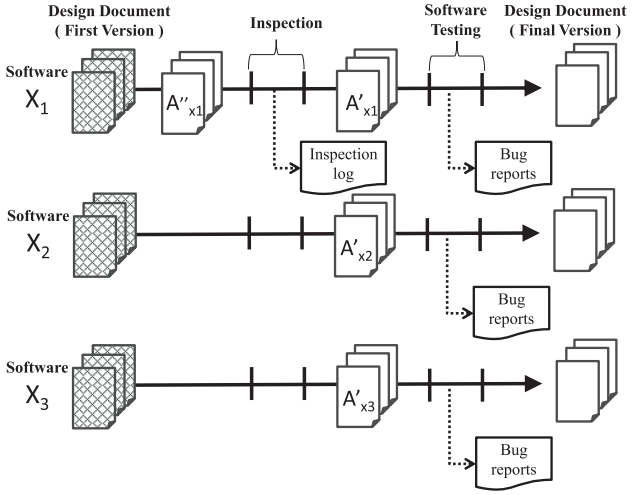


Fig. 5 Revisions of A''_x and A'_x.

3.2.2 Procedure

Evaluation 1.1: Evaluation 1.1 obtains the difference sets of single-words $\Delta'_s(R_x, A'_x)$ by Step 1 of the proposed approach. The inspector checks for omissions in the architectural design documents using $\Delta'_s(R_x, A'_x)$ by ensuring that the requirements containing one of the words in $\Delta'_s(R_x, A'_x)$ are implemented in the corresponding architectural design document. Evaluation 1.1 extracts single-words from the documents using the Japanese morphological analyzer MeCab [18], which extracts morphemes and their part of speech. Evaluation 1 regards nouns for single-words as the following reasons. First, a preliminary evaluation revealed that nouns characterize architectural design descriptions and that the number of different words of other parts of speech was smaller than that of nouns. Different words of other parts of speech were avoided to reduce ambiguity. Second, it is preferred that the number of words in $\Delta'_s(R_x, A'_x)$, $\Delta'_m(R_x, A'_x)$, and $\Delta'_h(R_x, A'_x)$ be small because the inspector's effort for the evaluation is limited.

Evaluation 1.2: Evaluation 1.2 obtains the difference set of multi-words $\Delta'_m(R_x, A'_x)$ by Step 2 of the proposed approach. The inspector checks for omissions in the architectural design documents using $\Delta'_m(R_x, A'_x)$ by ensuring that the specified requirements are implemented in the corresponding architectural design document.

Table 3 Number of single-words in $\Delta_s(R_x, A'_x)$ and $\Delta'_s(R_x, A'_x)$.

Document	Number of single-words in $\Delta_s(R_x, A'_x)$	Number of single-words in $C_F(\Delta_s(R_x, A'_x))$	Number of excluded low-frequent single-words in $\Delta_s(R_x, A'_x)$	Number of single-words in $\Delta'_s(R_x, A'_x)$
R_{x_1}, A''_{x_1}	753	78	646	29 *3
R_{x_1}, A'_{x_1}	699	57	613	29 *3
R_{x_2}, A'_{x_2}	215	161	-	54
R_{x_3}, A'_{x_3}	314	63	235	16 *3

Table 4 Most frequent single-words in $\Delta'_s(R_x, A'_x)$.

Document	Most frequent single-words in $\Delta'_s(R_x, A'_x)$
R_{x_1}, A''_{x_1}	Switching(129), Sheet(111), Modification(88), Direction(81), TX(33)
R_{x_1}, A'_{x_1}	Switching(129), Sheet(111), Modification(88), Direction(81), TX(33)
R_{x_2}, A'_{x_2}	Flow(12), Scope(6), Beforehand(6), Main(6), Exception(6)
R_{x_3}, A'_{x_3}	Case(44), Use(43), Series(17), Adjoining(5), Encapsulation(5)

Table 5 Detected $D(R_x, A'_x)$ and single-word triggers in $\Delta'_s(R_x, A'_x)$.

Document	$\Delta'_s(R_x, A'_x)$	$D(R_x, A'_x)$
R_{x_1}, A'_{x_1}	Switching	Switching function between a master device and a slave device.
R_{x_2}, A'_{x_2}	Priority	In the network connection, the connection priority is set to initiate a session.
R_{x_3}, A'_{x_3}	BIT	Specific BIT position in an output signal

Evaluation 1.3: Evaluation 1.3 obtains the difference set of word senses $\Delta'_h(R_x, A'_x)$ by Step 3 of the proposed approach. The inspector checks for omissions in the architectural design documents using $\Delta'_h(R_x, A'_x)$ by ensuring that the specified requirements are implemented in the corresponding architectural design document. Evaluation 1.3 uses Japanese WordNet [19] to identify whether a word has two or more word senses. Japanese WordNet defines 93,834 words and 158,058 word senses *2.

Evaluation 1.4: Evaluation 1.4 investigates whether single-words in $\Delta'_s(R_x, A'_x)$ and $C_F(\Delta_s(R_x, A'_x))$ are categorized into $C_i(\Delta_s(R_x, A'_x))$ according to certain features, attributes, or aspects. The inspector tries to categorize the single-words. The inspector also tries to categorize the multi-words in $\Delta'_m(R_x, A'_x)$ and $C_F(\Delta_m(R_x, A'_x))$ and the word senses in $\Delta'_h(R_x, A'_x)$ and $C_F(\Delta_h(R_x, A'_x))$.

3.3 Results

3.3.1 Evaluation 1.1

Table 3 lists the sizes of $\Delta_s(R_x, A'_x)$ and $\Delta'_s(R_x, A'_x)$. The numbers of single-words in $\Delta_s(R_{x_1}, A''_{x_1})$, $\Delta_s(R_{x_1}, A'_{x_1})$, and $\Delta_s(R_{x_3}, A'_{x_3})$ are all larger than 300. Due to the limited inspector's amount of effort for the evaluation, low-frequency single-words in $\Delta_s(R_{x_1}, A''_{x_1})$, $\Delta_s(R_{x_1}, A'_{x_1})$, and $\Delta_s(R_{x_3}, A'_{x_3})$ were excluded. Single-words whose appearance frequencies in the requirements specification document were less than five were excluded from the difference sets $\Delta_s(R_{x_1}, A''_{x_1})$, $\Delta_s(R_{x_1}, A'_{x_1})$, and $\Delta_s(R_{x_3}, A'_{x_3})$ before the inspector excluded the sets $C_F(R_x, A'_x)$. A total of 29 single-words exist in $\Delta'_s(R_{x_1}, A''_{x_1})$, and 29 single-words exist in $\Delta'_s(R_{x_1}, A'_{x_1})$, 54 single-words exist in $\Delta'_s(R_{x_2}, A'_{x_2})$, and 16 single-words exist in $\Delta'_s(R_{x_3}, A'_{x_3})$. **Table 4** lists the top five most frequent single-words in $\Delta'_s(R_x, A'_x)$. The single-words are translated into English because the documents are written in Japanese. In Table 4, the numbers in parentheses indicate the appearance frequencies of the single-words in R_x. In Evaluation 1.1, the inspector detected one omitted requirement categorized into $D(R_x, A'_x)$ from each of the three sets of documents (R_{x_1}, A'_{x_1}), (R_{x_2}, A'_{x_2}), and (R_{x_3}, A'_{x_3}). **Table 5** lists $D(R_x, A'_x)$ and the single-words in $\Delta'_s(R_x, A'_x)$ that were the detection triggers omitted requirements contained the single-words. **Table 6** shows the required effort for the inspector for each of the documents. Note that the evaluations for checking $\Delta_s(R_{x1}, A''_{x1})$ and $\Delta_s(R_{x1}, A'_{x1})$ are the same, because the words in $\Delta_s(R_{x1}, A''_{x1})$ and $\Delta_s(R_{x1}, A'_{x1})$ are the same. The sum of required effort for them is 7.3 person hour.

Table 6 Required effort (person-hour) for checking each of the words in difference set.

Document	$\Delta'(R_{x_1}, A''_{x_1})$	$\Delta'(R_{x_1}, A'_{x_1})$	$\Delta'(R_{x_2}, A'_{x_2})$	$\Delta'(R_{x_3}, A'_{x_3})$
single-word	7.3*	7.3*	13.5	4.0
multi-word	12.0*	12.0*	8.3	4.5

3.3.2 Evaluation 1.2

Table 7 lists the sizes of $\Delta_m(R_x, A'_x)$ and $\Delta'_m(R_x, A'_x)$. In Evaluation 1.2, due to inspector's effort limitation, low-frequency multi-words in $\Delta_m(R_{x_1}, A''_{x_1})$, $\Delta_m(R_{x_1}, A'_{x_1})$, and $\Delta_m(R_{x_3}, A'_{x_3})$ were also excluded. A total of 54 multi-words existed in $\Delta'_m(R_{x_1}, A''_{x_1})$, 48 multi-words existed in $\Delta'_m(R_{x_1}, A'_{x_1})$, 33 multi-words existed in $\Delta'_m(R_{x_2}, A'_{x_2})$, and 18 multi-words existed in $\Delta'_m(R_{x_3}, A'_{x_3})$. **Table 8** lists the top five most frequent multi-words in $\Delta'_m(R_x, A'_x)$. In Table 8, the multi-words are translated into English, and the numbers in parentheses indicate the appearance frequencies of the multi-words. In Evaluation 1.2, the inspector detected one omitted requirement categorized into $D(R_x, A'_x)$ from each of the three sets of documents (R_{x_1}, A'_{x_1}), (R_{x_2}, A'_{x_2}), and (R_{x_3}, A'_{x_3}). The requirements $D(R_x, A'_x)$ are the same requirements as detected with the single-words in $\Delta'_s(R_x, A'_x)$. **Table 9** shows $D(R_x, A'_x)$

*2 http://compling.hss.ntu.edu.sg/wnja/ (Wn-Ja1.1)

*3 Evaluation 1.1 obtains $\Delta'_s(R_x, A'_x)$ after excluding low-frequency single-words from $\Delta_s(R_x, A'_x)$

Table 7 Number of multi-words in $\Delta_m(R_x, A'_x)$ and $\Delta'_m(R_x, A'_x)$.

Document	Number of multi-words in $\Delta_m(R_x, A'_x)$	Number of multi-words in $C_F(\Delta_m(R_x, A'_x))$	Number of excluded low-frequency multi-words in $\Delta_m(R_x, A'_x)$	Number of multi-words in $\Delta'_m(R_x, A'_x)$
R_{x_1}, A''_{x_1}	1416	195	1173	48 *4
R_{x_1}, A'_{x_1}	1415	195	1172	48 *4
R_{x_2}, A'_{x_2}	301	268	-	33
R_{x_3}, A'_{x_3}	696	137	541	18 *4

Table 8 Most frequent multi-words in $\Delta'_m(R_x, A'_x)$.

Document	Most frequent multi-words in $\Delta'_m(R_x, A'_x)$
R_{x_1}, A''_{x_1}	Execute permission(82), Monitoring screen(39), Request signal(39), Status list(34), Configuration screen(34)
R_{x_1}, A'_{x_1}	Execute permission(82), Response signal(57), Monitoring screen(39), Request signal(39), Network switching(37)
R_{x_2}, A'_{x_2}	Error reply(9), Reliability(5), Device control(4), Control function(3), Status notification(3)
R_{x_3}, A_{x_3}	Mutual monitoring(5), Path switching(5), Time configuration(5), Maintenance function(5), Fault monitoring(5)

Table 9 Detected $D(R_x, A'_x)$ and multi-word triggers in $\Delta'_m(R_x, A'_x)$.

Document	$\Delta'_m(R_x, A'_x)$	$D(R_x, A'_x)$
R_{x_1}, A'_{x_1}	Switching function	Switching function between a master device and a slave device.
R_{x_2}, A'_{x_2}	Connection priority	In the network connection, the connection priority is set to initiate a session.
R_{x_3}, A'_{x_3}	BIT position	Specific BIT position in an output signal

and the multi-words in $\Delta'_m(R_x, A'_x)$ that were their detection triggers. Table 6 shows the required effort for the inspector for each of the documents. As noted in Section 4.1.1, the evaluations for checking $\Delta'_m(R_{x1}, A''_{x1})$ and $\Delta'_m(R_{x1}, A'_{x1})$ are the same, because the words in $\Delta'_m(R_{x1}, A''_{x1})$ and $\Delta'_m(R_{x1}, A'_{x1})$ are the same. The sum of required effort for them is 12 person hour.

3.3.3 Evaluation 1.3

In Evaluation 1.3, the inspector could not identify any single-word that had multiple senses in R_x. Each of the single-words was defined as a homonymous word in Japanese WordNet, however, no single-words had two or more senses in R_x. **Table 10** shows the number of the single-words in $w_h(R_x, A'_x)$.

3.3.4 Evaluation1.4

In Evaluation 1.4, the inspector categorized single-words in $\Delta'_s(R_x, A'_x)$ into C_1 and C_2 and categorized single-words in $C_F(\Delta_s(R_x, A'_x))$ into C_3 and C_4. The inspector also categorized the multi-words in $\Delta'_m(R_x, A'_x)$ and $C_F(\Delta_m(R_x, A'_x))$ into the same C_1, C_2, C_3, and C_4 categories defined for the single-words. As mentioned in previous subsection, due to the limitation of the inspector's effort, single-words and multi-words whose appearance frequencies were less than five in $\Delta_s(R_x, A'_x)$ and $\Delta_m(R_x, A'_x)$ were excluded. The categories are shown as follows:

- C_1: words for functional requirements such as detection, flow, error response, and bandwidth limitations
- C_2: words for non-functional requirements such as availability, security, and redundant architecture
- C_3: words irrelevant to software design such as CPU, memory, physical port, and transmit frequency, and
- C_4: words for the development project such as version, document, standard, and normal case.

Tables 11 and **12** list the number of single-words and multi-words, respectively, in the categories. Two or more single-words in $\Delta_s(R_{x_1}, A''_{x_1})$, $\Delta_s(R_{x_1}, A'_{x_1})$, and $\Delta_s(R_{x_2}, A'_{x_2})$ were categorized into each of the categories C_1, C_2, C_3, and C_4. Two or more single-words in $\Delta_s(R_{x_3}, A'_{x_3})$ were categorized into each of the categories C_1, C_3, and C_4. Two or more multi-words in $\Delta_m(R_{x_1}, A'_{x_1})$ and $\Delta_m(R_{x_2}, A'_{x_2})$ were categorized into each of the categories C_1, C_2, C_3, and C_4. Two or more multi-words in $\Delta_m(R_{x_1}, A''_{x_1})$ and $\Delta_m(R_{x_3}, A'_{x_3})$ were categorized into each of the categories C_1, C_3,

Table 10 Number of words defined as homonymous words in Japanese WordNet.

Document	Number of homonymous words
R_{x_1}, A''_{x_1}	173
R_{x_1}, A'_{x_1}	173
R_{x_2}, A'_{x_2}	90
R_{x_3}, A'_{x_3}	120

Table 11 Number of single-words in $\Delta'_s(R_x, A'_x)$ and $C_F(\Delta_s(R_x, A'_x))$ in each category.

Document	C_1	C_2	C_3	C_4
R_{x_1}, A''_{x_1} *5	26	3	19	42
R_{x_1}, A'_{x_1} *5	27	2	17	35
R_{x_2}, A'_{x_2}	51	3	70	70
R_{x_3}, A'_{x_3} *5	16	0	10	40

Table 12 Number of multi-words in $\Delta'_m(R_x, A'_x)$ and $C_F(\Delta_m(R_x, A'_x))$ in each category.

Document	C_1	C_2	C_3	C_4
R_{x_1}, A''_{x_1} *6	48	0	3	132
R_{x_1}, A'_{x_1} *6	45	3	11	133
R_{x_2}, A'_{x_2}	25	8	39	198
R_{x_3}, A'_{x_3} *6	18	0	6	105

*4 Evaluation 1.2 obtains $\Delta'_m(R_x, A'_x)$ after excluding low-frequency multi-words from $\Delta_m(R_x, A'_x)$.

*5 Evaluation 1.4 categorizes $\Delta'_s(R_x, A'_x)$ and $C_F(\Delta_s(R_x, A'_x))$ after excluding low-frequency single-words from $\Delta_s(R_x, A'_x)$.

*6 Evaluation 1.4 categorizes $\Delta'_m(R_x, A'_x)$ and $C_F(\Delta_m(R_x, A'_x))$ after excluding low-frequency multi-words from $\Delta_m(R_x, A'_x)$.

Table 13 Categories for single-words in Table 4.

	C_1	C_2	C_3	C_4
x_1	Switching		Direction, TX	Sheet, Modification
x_2			Scope, Beforehand	Flow, Main, Exception
x_3			Adjoining, Encapsulation	Case, Use, Series

Table 14 Categories for multi-words in Table 8.

	C_1	C_2	C_3	C_4
$x_1(A''_{x_1})$	Request signal, Status list, Configuration screen		Execute permission, Monitoring screen	Status list
$x_1(A'_{x_1})$	Request signal, Execute permission, Response signal		Monitoring screen, Network switching	
x_2	Error reply, Device control, Control function, Status notification	Reliability		
x_3	Path switching, Maintenance function, Fault monitoring		Mutual monitoring, Time configuration	

and C_4. Evaluation 1.4 was not conducted for the difference set of word senses $\Delta_h(R_x, A'_x)$ because Evaluation 1.3 found no multiple word senses in $\Delta_h(R_x, A'_x)$.

Omissions of requirements including words categorized into C_1 and C_2 lead to omissions in architectural design document. Omissions of requirements including words in C_3 and C_4 do not indicate an omissions in architectural design document. Omitted words in C_3 may appear in another document. Omitted words in C_4 are not used to describe requirements. Omissions of words in C_1 indicate lack or insufficient function definitions. Omissions of words in C_2 indicate lack or insufficient non-functional definitions such as performance and efficiency. If the omitted words are replaced with other words, omissions of words in C_1 and C_2 do not indicate omissions in the architectural design document. As shown in Table 5 and **Table 13**, the word "switching" is categorized as C_1 and indicates that the omission is in the architectural design document. The words "TX" and "direction" are categorized as C_3. The word TX is included in the requirement defining an operation mode. The word direction is included in the requirement defining direction of communication. These requirements are implemented in other documents. The words "sheet" and "modification" are categorized as C_4. These words are included in the update history of the requirement specification document.

Table 13 and **Table 14** show categories of the single-words and multi-words in Tables 4 and 8.

3.4 Evaluation 2

3.4.1 Materials

The materials in Evaluation 2 included six pairs of requirements specification documents and the corresponding architectural design documents of systems y_1, y_2, y_3, y_4, y_5, and y_6, as well as the three pairs of documents used in Evaluation 1. The pairs of documents R_{y_1} and A_{y_1}, R_{y_2} and A_{y_2}, R_{y_3} and A_{y_3}, and R_{y_4} and A_{y_4} were written in Japanese. The pairs of documents R_{y_5} and A_{y_5}, and R_{y_6} and A_{y_6} were written in English. Documents R_{y_1}, R_{y_2}, R_{y_3}, R_{y_5}, R_{y_6}, A_{y_1}, A_{y_2}, A_{y_3}, A_{y_5}, and A_{y_6} were available online. Documents R_{y_1}, R_{y_2}, R_{y_3}, A_{y_1}, A_{y_2}, and A_{y_3} were the artifacts of outsourced software development by public organizations including a public library and a disaster response organization. Documents R_{y_4} and A_{y_4} resulted from an outsourced software tool development project for academic research. The pairs R_{y_5} and A_{y_5}, and R_{y_6} and A_{y_6} were the artifacts of NASA development projects [14]. These pairs were publicly available as a software engineering repository dataset online [15], [16], [17]. **Table 15** lists the sizes of the documents. Because documents R_{y_5}, R_{y_6}, A_{y_5}, and A_{y_6} are only available as plain text files, their numbers of pages are left blank in Table 15. **Table 16** lists the groups of the materials.

Table 15 Sizes of the documents R_y and A_y.

Document	Number of pages	Number of single-words	Number of multi-words	Number of letters
R_{y_1}	45	1088	1911	44101
R_{y_2}	12	481	473	8038
R_{y_3}	75	1293	1459	45879
R_{y_4}	5	72	55	1152
R_{y_5}	-	269	145	26910
R_{y_6}	-	87	41	3495
A_{y_1}	38	1273	2383	42490
A_{y_2}	58	851	849	26610
A_{y_3}	136	1285	2497	76039
A_{y_4}	14	206	191	3988
A_{y_5}	-	703	561	89475
A_{y_6}	-	86	26	4654

3.4.2 Procedure

Evaluation 2 investigates the ratios of the numbers of single-words in the difference sets $\Delta_s(R, A)$ and $\Delta_m(R, A)$ to the number of words in the nine requirements specification documents. The architectural design documents in six of the pairs were final versions revised for the correction of defects detected in the inspections and testing. The ratios of the numbers of single-word are defined as follows.

$$Ratio_s(R, A) = \frac{|\Delta_s(R, A)|}{|W_s(R)|}$$

Table 16 Groups of the materials.

Group	Pair of documents	Explanation
$G_A(R, A')$	(R_{x_1}, A''_{x_1}), (R_{x_1}, A'_{x_1}),(R_{x_2}, A'_{x_2}), (R_{x_3}, A'_{x_3})	The same documents as in Evaluation 1
$G_B(R, A)$	(R_{y_1}, A_{y_1}), (R_{y_2}, A_{y_2}), (R_{y_3}, A_{y_3}), (R_{y_4}, A_{y_4})	Documents y_1, y_2, and y_3 were publicly available. All documents were written in Japanese.
$G_C(R, A)$	(R_{y_5}, A_{y_5}), (R_{y_6}, A_{y_6})	All documents were publicly available and written in English.

Table 17 $Ratio_s(R, A)$ in document pairs (R_x, A'_x) and (R_y, A_y).

Group	Document	$\lvert\Delta_s(R, A)\rvert$	$\lvert W_s(R)\rvert$	$Ratio_s(R, A)$
G_A	R_{x_1}, A''_{x_1}	753	1318	0.57
	R_{x_1}, A'_{x_1}	699	1318	0.53
	R_{x_2}, A'_{x_2}	215	503	0.43
	R_{x_3}, A'_{x_3}	314	663	0.47
G_B	R_{y_1}, A_{y_1}	414	1088	0.38
	R_{y_2}, A_{y_2}	113	481	0.23
	R_{y_3}, A_{y_3}	747	1293	0.58
	R_{y_4}, A_{y_4}	13	72	0.18
G_C	R_{y_5}, A_{y_5}	78	269	0.29
	R_{y_6}, A_{y_6}	57	87	0.65

Table 18 $Ratio_m(R, A)$ in document pairs (R_x, A'_x) and (R_y, A_y).

Group	Document	$\lvert\Delta_m(R, A)\rvert$	$\lvert W_m(R)\rvert$	$Ratio_m(R, A)$
G_A	R_{x_1}, A''_{x_1}	1416	1889	0.75
	R_{x_1}, A'_{x_1}	1415	1889	0.75
	R_{x_2}, A'_{x_2}	301	481	0.63
	R_{x_3}, A'_{x_3}	696	974	0.71
G_B	R_{y_1}, A_{y_1}	1505	1911	0.79
	R_{y_2}, A_{y_2}	227	473	0.48
	R_{y_3}, A_{y_3}	1134	1459	0.78
	R_{y_4}, A_{y_4}	20	55	0.36
G_C	R_{y_5}, A_{y_5}	102	145	0.70
	R_{y_6}, A_{y_6}	34	41	0.83

$$Ratio_m(R, A) = \frac{|\Delta_m(R, A)|}{|W_m(R)|}$$

Evaluation 2 also investigates the distributions of the appearance frequencies of single-words in R_x and R_y. Evaluation 2 extracts words from group $G_C(R, A)$ using the English morphological analyzer TreeTagger [20].

3.5 Results

Table 17 lists the $Ratio_s(R, A)$ in the document groups $G_A(R, A')$, $G_B(R, A)$, and $G_C(R, A)$.

Table 18 lists the $Ratio_m(R, A)$ in the document groups $G_A(R, A')$, $G_B(R, A)$, and $G_C(R, A)$. The numbers of multi-words in $\Delta_m(R, A')$ and $\Delta_m(R, A)$ are larger than the numbers of single-words in all pairs of documents in all document groups. In addition, $Ratio_m(R, A)$ are larger than $Ratio_s(R, A)$ in all document groups. The Pearson correlation coefficient between $\Delta_s(R, A)$ and $W_s(R)$ is 0.970 ($p = 0.0079$), and that between $\Delta_m(R, A)$ and $W_m(R)$ is 0.997 ($p = 0.00018$). **Figures 6–8** show the distributions of the appearance frequencies of the single-words in the difference set $\Delta_s(R, A)$ in the requirements specification document R for the document groups of $G_A(R, A')$, $G_B(R, A)$, and $G_C(R, A)$. In Figs. 6–8, the horizontal axes represent the appearance frequencies of the single-words in the requirements specification document R and the vertical axes represent the percentages of the single-words. The white bars represent distributions of appearance frequencies of the single-words in $W_s(R)$. The percentages are the numbers of single-words of the corresponding appearance frequencies indicated horizontal axes to the numbers of single-words in $W_s(R)$.

4. Discussion

4.1 Research Question 1

Our results indicate that an inspector can indeed detect omissions in an architectural design document by checking the difference set between words in the requirements specification document and words in the corresponding architectural design document. Evaluation 1.1 also indicated that an independent inspector who was not involved in the specific project could detect omissions using the proposed approach.

In Evaluation 1.2, the detected omissions were the same as the omissions detected in Evaluation 1.1. A discussion with the inspector revealed that multi-words could enable inspectors with less experience to detect omissions that could not be detected with single-words because multi-words are more specific than single-words. Future studies will include investigations on detecting omissions with single-words and multi-words with inspectors with different levels of experience. In addition, future studies will include reducing the number of multi-words by modifying the algorithm used to extract them.

As shown in Tables 5 and 9, each pair of single-word and multi-word indicated the same potential requirement omission. For system x_1, the single-word "switching" and multi-word "switching function" were omitted. Requirement specification document R_{x_1} defines a requirement for redundant communication network devices and a switching function using the words "switching function". The function switches master and slave communication network devices in the case of the master device failure. However, the architectural design document A'_{x_1} does not define preconditions and postconditions of switching and switching behaviors. The reason for the insufficient definition is that the designer did not understand the demand for switching. In Evaluations 1.1 and 1.2, each of the three pairs of the requirement specification document and the architectural design document has its own difference sets of single-words and multi-words. If all words in the requirement specification documents are included in the corresponding architectural design document, the proposed approach cannot help designers and inspectors detect omissions. In such cases, incorrect and insufficient definitions can exist in the architectural design document. To clarify the limitation of the proposed approach, investigating the domains, characteristic of documents, and criterions for such cases is one of the important future studies.

For system x_2, the single-word "priority" and multi-word "connection priority" were omitted. Requirement specification document R_{x_2} defines a requirement for the priority of receiving healthiness information from network monitoring devices in the

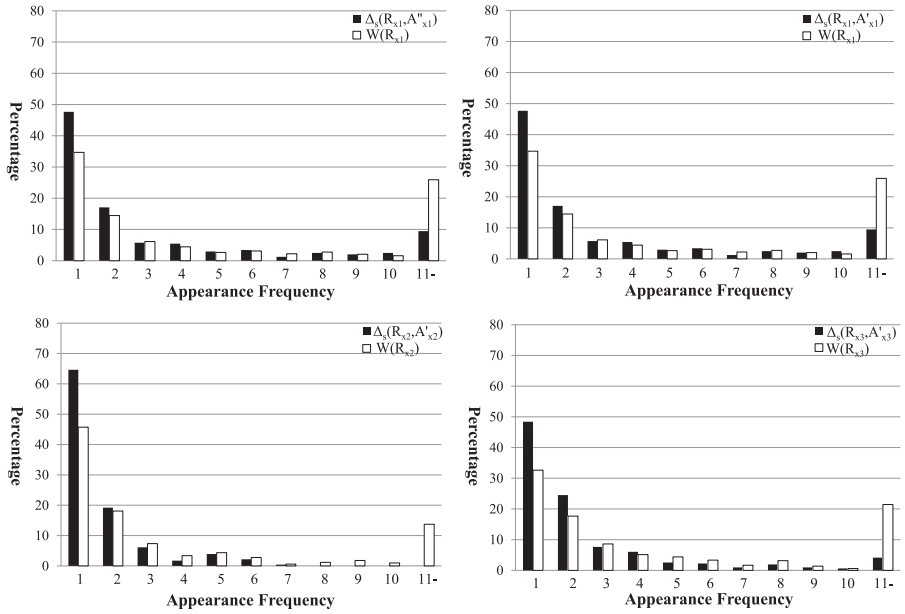


Fig. 6 Distribution of $W_s(R)$ and $\Delta_s(R, A)$ in documents group G_A.

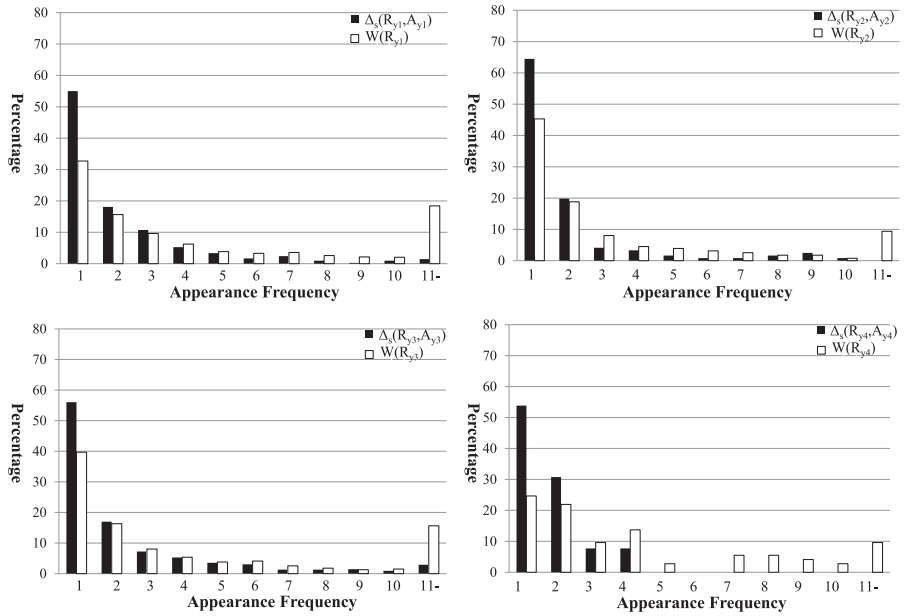


Fig. 7 Distribution of $W_s(R)$ and $\Delta_s(R, A)$ in documents group G_B.

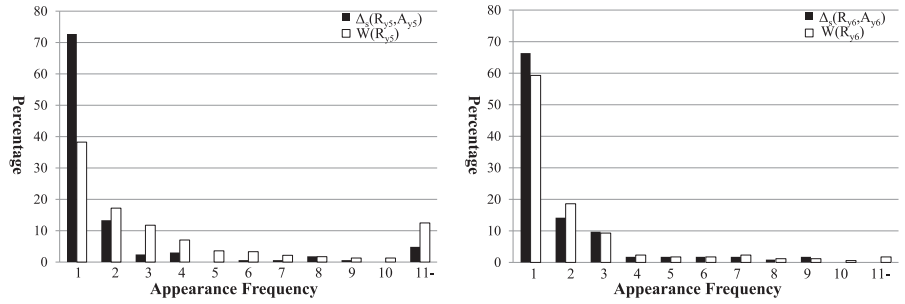


Fig. 8 Distribution of $W_s(R)$ and $\Delta_s(R, A)$ in documents group G_C.

case that two or more devices try to connect the control system. However, architectural design document A'_{x_2} does not define a procedure for determining priorities of the network monitoring devices. The reason for the insufficient definition is that the designer did not presume the two or more connections at the same time. If this omission is not corrected, network failure detection may be delayed. If a low-priority monitoring device does not disconnect the connection to the control system, a high-priority monitoring device must wait for timeout. The control system may not receive failure information immediately. Connection priority

affects effectiveness and reliability quality attributes defined in ISO 25010. Furthermore, the increase of the monitoring devices in future system evolutions raises the possibility of multiple connections, the omission of connection priority affects expandability quality attributes. If a malicious client connects to the control system and disconnects the connection, the client may attack the control system by performing denial-of-service. This may affect the security quality attribute defined by ISO 25010.

For system x_3, the single-word "BIT" and multi-word "BIT position" were omitted. Requirement specification document R_{x_3} defines a requirement for defining the bit position representing a network connection failure data in a network data frame. However, architectural design document A'_{x_3} does not explicitly define the position of the bit. The reason for the insufficient definition is that the designer did not suppose that the bit positions between sender and receiver can be different depending on sending and receiving devices. In the case that this omission is not corrected, the communication network failure data cannot be received. This may disable detecting communication network failure. If the bit position is not defined in the architectural design document, it may have an impact on the effectiveness and the reliability quality attributes of the system. Furthermore, the omission may affect expandability because the monitoring devices that have different encoding procedures may be connected in the future system evolutions.

System y_1 is a city public library system. For system y_1, the single-word "compensation" and multi-word "compensation work" were omitted. Requirement specification document R_{y_1} defines the printing function for compensation notice. The requirements are implemented in another design document rather than in architectural design document A_{y_1}. Omitting this requirement affects effectiveness and reliability quality attributes because omitting the requirement may increase unreturned books.

System y_2 is a disaster prevention system. The single-word "restore" was omitted. Requirement specification document R_{y_2} defines a requirement for defining file backup and restore functions. Architectural design document A_{y_2} defines the corresponding function using the words "recovery." Omitting this function affects the reliability quality attribute. Furthermore, omitting this function affects security and expandability because this function may cause data loss, which increases the possibility of malicious attacks and the system evolution difficulty.

System y_3 is a housing history management system. The single-word "protection" was omitted. Requirement specification document R_{y_3} defines security requirements. Architectural design document A_{y_3} describes security functions using the words "leakage protection," and "data corruption protection." Omitting the security function affects reliability and security quality attributes.

System y_4 is an assurance case development system. The single-word "generation" was omitted. Requirement specification document R_{y_4} defines a requirement defining a function that generates assurance case properties. Omitting the function affects the reliability and functional correctness quality attributes because the properties may be insufficient to argue the goal such as safety in the assurance case.

System y_5 is a measurement system. The single-word "bounds" and multi-word "bounds checking" were omitted. Requirement specification document R_{y_5} defines a requirement performing bounds checking before accessing memory. Architectural design document A_{y_5} describes a corresponding function using the word "the largest free memory block." Omitting the function affects the reliability quality attribute because the function prevents the buffer overflow.

System y_6 is an infrared radiation system attached to an earth observation satellite. The single-word "core" and multi-word "core metadata" were omitted. Requirement specification document R_{y_6} defines a requirement containing the metadata of the environmental control subsystem. Architectural design document A_{y_6} describes the corresponding function using the words "standard" and "standard global metadata." Omitting the requirement may affect the reliability quality attribute.

In Evaluations 1.1 and 1.2, the inspector checked each word in the difference sets one by one by keyword search to accurately evaluate the proposed approach. However, in real situations, inspectors do not always need to check each word individually nor to use keyword search in the case that the inspector has confidence. They can glance at the list of words to detect omissions to reduce the effort required for the proposed approach.

Evaluation 1.3 could not find words having two or more word senses among the possible homonymous words extracted by the proposed approach in all pairs of documents. The proposed approach uses a dictionary to identify words with two or more word senses, however, in the documents, only one sense of those defined in the dictionary was used. A more specific dictionary for software documents may lead to identifying additional homonymous words.

In Evaluation 1.4, words categorized into C_2 were not found among single-words and multi-words whose appearance frequencies in the requirements specification documents were more than or equal to 5 among two pairs of requirements specification documents and architectural design documents. The inspector categorized all of the sinle-words in $\Delta_s(R_{x_2}, A'_{x_2})$ and the multi-words in $\Delta_m(R_{x_2}, A'_{x_2})$. The percentage of the single-words categorized into C_3 or C_4 to the single-words in $\Delta_s(R_{x_2}, A'_{x_2})$ was 72% and that of the multi-words to the multi-words in $\Delta_m(R_{x_2}, A'_{x_2})$ was 82%. By using the categories as a guide, inspectors can quickly identify whether a word is categorized into $\Delta'_s(R, A)$ or $C_F(\Delta_s(R, A))$. In software maintenance and evolutions, filtering words by tool may lead to semi-automatic identification. The tool stores the words that have already been categorized as C_1, C_2, C_3, and C_4 in the prior versions and categorizes words extracted in a subsequent version into the categories using the stored words.

General reasons for omissions of words that are categorized as C_1 and C_2 and not replaced with other words in the corresponding architectural design document are as follows. First, requirements including the omitted words are not fully specified. Second, designers and inspectors misunderstand the requirements and define insufficient design descriptions to cover the requirements. Designers and inspectors should focus on detecting omitted requirements including words that are categorized as C_1 and C_2 and replaced with other words in an architectural design document. The target systems in Evaluation 1 are communication network moni-

toring systems. Words indicating communication network failure detection and recovery have higher priorities to check because the risks of omissions of communication network failure detection are more severe.

Evaluations 1.1 and 1.2 obtained omitted single-words and multi-words by comparing requirement specification documents R_{x_1}, R_{x_2}, and R_{x_3} and architectural design documents A''_{x_1}, A'_{x_1}, A'_{x_2}, and A'_{x_3}. As shown in Fig. 5, architectural design document A''_{x_1} is the version before the design inspection. Architectural design documents A'_{x_1}, A'_{x_2}, and A'_{x_3} are the versions after correcting the defects detected in the design inspections. If defects were detected in subsequent development activities to the design inspection such as testing, the architectural design documents were changed. For example, the requirements, including the word "switching" in Table 5, are not sufficiently defined in the architectural design document A'_{x_1}. Evaluation 1 uses the intermediate versions of architectural design documents A''_{x_1}, A'_{x_1}, A'_{x_2}, and A'_{x_3} because the architectural design documents may include corrected defects after design inspections. Therefore, Evaluation 1 can investigate whether the proposed approach can identify the potentially omitted requirements. Evaluation 2 uses the final versions of the architectural design documents A_{y_1}, A_{y_2}, ... , A_{y_6}, because the intermediate versions of the architectural documents are not available. Since the architectural design documents are the versions after testing, they are supposed to include no word categorized as C_1 or C_2 without replacement with other words in the architectural design documents.

4.2 Research Question 2

The number of words that need to be checked when using the proposed approach depends on the number of words in the requirements specification document. Inspectors can estimate the number of words in the difference set by measuring the number of words in the requirements specification document and deciding whether to use the proposed approach.

Approximately half of the words in the difference sets have the potential to be excluded by the semi-automatic identification approach mentioned above, because the percentages of the number of words in $C_F(\Delta_s(R, A))$ to the numbers of words in the difference sets were larger than 45%.

The percentages of low appearance frequency single-words in $\Delta_s(R, A)$ are larger than those of $W_s(R)$. Evaluations 1.1, 1.2, and 1.4 excluded words whose appearance frequency were less than five from three of the four difference sets. The percentage of the number of the excluded words were larger than 70%. It is possible that the inspector can detect omissions in the architectural design documents by checking the excluded words. Future studies will include investigations on detecting omissions with low-frequency words and reducing or prioritizing the words.

4.3 Threats to Validity

Evaluation 1 was conducted in different settings from those of a real inspection. This might be a threat to the approach's external validity. The materials in Evaluation 1 were developed in the past development. Evaluation 1 was conducted without time restrictions. However, all of the architectural design documents were intermediate versions that were modified to correct detected defects in the design inspection or testing phase. Discussions with the inspector revealed that inspectors involved in the development could probably detect defects using the proposed approach and that the required level of effort could be acceptable in real inspections under the assumption that the number of words is smaller than 300. Future studies will include evaluations in real inspection settings.

Evaluations 1.1 and 1.2 excluded low-frequency words from the difference sets in two of the three pairs of documents. Excluding low-frequency words from the difference sets does not affect the internal validity of the evaluation because the evaluations were conducted with the remaining words. An inspector might be able to detect additional omissions with the excluded words. Future studies will include investigations on the exclusion criteria and comparisons to difference sets without exclusions.

In Evaluations 1.1 and 1.2, each of the three pairs of requirement specification document and architectural design document has its own difference sets of single-words and multi-words. If all words in the requirement specification documents are included in the corresponding architectural design document, the proposed approach cannot help designers and inspectors detect omissions. In such cases, incorrect and insufficient definitions can exist in the architectural design document. To clarify the limitation of the proposed approach, investigating the domains, characteristic of documents, and criterions for such cases is one of the important future studies.

In Evaluations 1.1 and 1.2, we gave up obtaining recall metrics of detected omissions, although F measure (harmonic mean of precision and recall) could show the effectiveness of the proposed approach. In Evaluation 1, the recalls can be obtained by calculating the percentages of the number of detected omissions by the proposed approach to the number of detected omission defects by the design inspections, which we set as the baseline. However, in the defect lists in the target systems, the defect classification did not include omission, so we need to check whether each of the defects recorded in the defect lists is an omission defect or not. The defect lists were recorded in the design inspections and in the design inspections and the testing phase. The total number of defects in the defect lists in Evaluation 1 was over 600. The effort for obtaining the recalls is too large for Evaluation 1. Evaluations of the proposed approach by F measure is one of the important future works.

In Evaluation 2, the selection and representativeness of the materials might be a threat to the approach's external validity. The additional materials were five pairs of documents that were available online and one pair for documents of a research tool for in-house use. However, the publicly available materials were the artifacts of public procurement software development projects. Seven of the pairs of documents written in Japanese contained standard items defined by standards such as IEEE 830, and the development activities had been managed by following standards based on PMBOK [21]. Therefore, the documents are expected to be representative of documents including such standard items under development management following such standards.

5. Related Work

To reduce the effort for manual inspection, several approaches have been developed to support inspectors in validating software documents with natural language processing. The tool REquirements TRacing On-target (RETRO) [22] supports inspectors by indicating traceability links between sentences in a requirements specification document and the corresponding sentences in an architectural design document. This tool extracts requirement sentences from a requirements specification document and design sentences from the corresponding architectural design document. Then, it calculates the similarities in every pair of sentences. The tool suggests candidate traceability links between the requirement sentences and the design sentences to the developers. The candidate traceability links are selected by the highest similarities. These approaches are different from the proposed approach because the proposed approach does not identify traceability links between sentences.

To reduce the effort required by checklist-based source code reviews, there are several approaches to support maintaining such checklists. The tool Program Assurance Environment (PAE) [23] ensures consistency and detects redundancy in checklist items using natural language processing. The tool Collaborative Software Inspection (CSI) [24] reduces the effort required by the inspection coodinator to summarize defects by integrating the defect lists from individual inspectors. These approaches aim to provide an integrated software inspection environment, however, they are different from the proposed approach because they do not analyze artifacts.

Some approaches automatically detect potential defects from artifacts [7], [9]. The tool Intelligent Code Inspection in a C Language Environment (ICICLE) [7] supports inspectors validating the source code written in C or C++. This tool detects potential defects such as coding rule violations by predefined criteria. These approaches are different from the proposed approach because they require predefined criteria.

In addition, there are approaches that predict defect-prone locations using source code metrics as input [8], [25], [26]. These approaches use prediction models such as logistic regression and Bayesian models, however, they are different from the proposed approach because they require source code metrics and use prediction models.

6. Conclusion

This article proposes an approach to detect omitted requirements that are not implemented in the corresponding architectural design document using difference sets of words or word senses between a software requirements specification document and the corresponding software architectural design document. First, the proposed approach extracts sets of single-words, multi-words, and word senses that appear in a software requirements specification document and do not appear in the corresponding software architectural design document by using a natural language processing tool. Then, an architectural design document inspector validates whether each of the requirements specified with single-words, multi-words, or word senses are implemented in the corresponding architectural design document using the set as guides.

To investigate the effectiveness and efficiency of the proposed approach, we conducted two evaluations. Evaluation 1 investigated whether omitted requirements can be detected in architectural design documents using the proposed approach. The result of Evaluation 1 shows that omitted requirements were detected in all three pairs of software requirements specification documents and architectural design documents investigated.

Evaluation 2 investigated the number of words that inspectors need to check in the proposed approach. The result of Evaluation 2 shows that the percentages of different single-words in the difference sets to different single-words that appear in the requirements specification documents vary from 18 to 65 for the nine pairs of software requirements specification documents and architectural design documents investigated. The result also shows that the percentages of different multi-words in the difference sets to the different multi-words that appear in the requirements specification documents vary from 36 to 83. Pearson correlation coefficient between the number of single-words in the difference set and the number of single-words in the requirements specification documents is 0.970 ($p = 0.0079$). Pearson correlation coefficient between the number of multi-words in the difference set and the number of multi-words in the requirements specification documents is 0.997 ($p = 0.0018$).

Acknowledgments This work was supported by JSPS KAKENHI Grant Number JP17H00731.

References

[1] Ebert, C. and Jones, C.: Embedded Software: Facts, Figures, and Future, *IEEE Computer*, Vol.42, No.4, pp.42–52 (2009).
[2] Fagan, M.: Design and code inspections to reduce errors in program development, *Software Pioneers*, pp.575–607, Springer (2002).
[3] Shull, F., Rus, I. and Basili, V.R.: How perspective-based reading can improve requirements inspections, *Computer*, Vol.33, No.7, pp.73–79 (2000).
[4] Porter, A.A., Votta, L.G. and Basil, V.R.: Comparing detection methods for software requirements inspections: A replicated experiment, *IEEE Trans. Software Engineering*, Vol.21, No.6, pp.563–575 (1995).
[5] Basili, V.R.: Software modeling and measurement: The goal/question/metric paradigm, Technical report (1992).
[6] Mashiko, Y. and Basili, V.R.: Using the GQM paradigm to investigate influential factors for software process improvement, *Journal of Systems and Software*, Vol.36, No.1, pp.17–32 (1997).
[7] Brothers, L., Sembugamoorthy, V. and Muller, M.: ICICLE: Groupware for code inspection, *Proc. 1990 ACM Conference on Computer-supported Cooperative Work*, pp.169–181 (1990).
[8] Wilbur, W.J. and Sirotkin, K.: The automatic identification of stop words, *Journal of information science*, Vol.18, No.1, pp.45–55 (1992).
[9] Sembugamoorthy, V. and Brothers, L.: ICICLE: Intelligent code inspection in a C language environment, *Computer Software and Applications Conference, COMPSAC 90, Proc. 14th Annual International*, pp.146–154 (1990).
[10] Wakabayashi, T., Morisaki, S., Atsumi, N. and Yamamoto, S.: An empirical evaluation of detecting omissions by comparing words between requirement and architectural documents, *Proc. 8th International Workshop on Empirical Software Engineering in Practice* (*IWESEP*), pp.12–17 (2017).
[11] Weiss, S.M., Indurkhya, N., Zhang, T. and Damerau, F.: *Text mining: Predictive Methods for Analyzing Unstructured Information*, Springer Science & Business Media (2010).
[12] Runeson, P., Alexandersson, M. and Nyholm, O.: Detection of duplicate defect reports using natural language processing, *Proc. 29th International Conference on Software Engineering*, pp.499–510 (2007).
[13] Kitchenham, B., Pickard, L. and Pfleeger, S.L.: Case Studies for Method and Tool Evaluation, *IEEE Software*, Vol.12, No.4, pp.52–62 (1995).

[14] Hayes, J.H., Sundaram, S.K. and Dekhtyar, A.: Baselines in Requirements Tracing, International Workshop on Predictor Models in Software Engineering St. Louis (2005).
[15] Shirabad, S.J. and Menzies, T.J.: The PROMISE Repository of Software Engineering Databases, School of Information Technology and Engineering, University of Ottawa, Canada (2005).
[16] MDP Website, CM-1 Project (2005), available from ⟨http://mdp.ivv.nasa.gov/mdp_glossary.html#CM1⟩.
[17] MODIS Science Data Processing Software Requirements Specification Version 2 (1997).
[18] Kudo, T.: Mecab: Yet another part-of-speech and morphological analyzer (2005), available from ⟨http://mecab.sourceforge.net/⟩.
[19] Isahara, H., Bond, F., Uchimoto, K., Utiyama, M. and Kanzaki, K.: Development of the Japanese WordNet (2008).
[20] Schmid, H.: Treetagger— A language independent part-of-speech tagger, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Vol.43, p.28 (1995).
[21] Pm, I.: *A guide to the project management body of knowledge* (*PMBOK guide*), Project Management Institute (2000).
[22] Hayes, J.H., Dekhtyar, A. and Sundaram, S.K.: Advancing candidate link generation for requirements tracing: The study of methods, *IEEE Trans. Software Engineering*, Vol.32, No.1, pp.4–19 (2006).
[23] Belli, F. and Crisan, R.: Towards automation of checklist-based code-reviews, *Proc. 7th International Symposium on Software Reliability Engineering*, pp.24–33 (1996).
[24] Mashayekhi, V., Drake, J.M., Tsai, W.-T. and Riedl, J.: Distributed, collaborative software inspection, *IEEE Software*, Vol.10, No.5, pp.66–75 (1993).
[25] Fenton, N.E. and Ohlsson, N.: Quantitative analysis of faults and failures in a complex software system, *IEEE Trans. Software Engineering*, Vol.26, No.8, pp.797–814 (2000).
[26] Menzies, T., Greenwald, J. and Frank, A.: Data mining static code attributes to learn defect predictors, *IEEE Trans. Software Engineering*, Vol.33, No.1, pp.2–13 (2007).

Takehiro Wakabayashi is in the Graduate School of Information Science Nagoya University, Japan. He obtained his Bachelor of Engineering degree from the Department of Information Engineering, School of Engineering, Nagoya University, Japan in 2016. His research interests include empirical software engineering and software reviews and inspections.

Shuji Morisaki is an associate professor at Nagoya University, Japan. Previously, he was a software engineer in the Japanese software industry. He received his Doctor of Engineering degree from Graduate School of Information Science, Nara Institute of Science and Technology, Japan in 2001. His research interests include empirical software engineering, software reviews and inspections, and mining software repositories.

Norimitsu Kasai is a system engineer at Mitsubishi Electronic Corporation. He received his Doctor of Engineering degree from Nara Institute of Science and Technology, Japan in 2014. His research interests include software / system engineering, and source code analysis.

Noritoshi Atsumi is a designated assistant professor of Strategy Office, Information and Communications, Nagoya University. He received Doctor of Engineering from Nagoya University in 2007. His research interests include software reuse, software development support, software maintenance and program analysis. He is a member of JSSST, IEICE, IPSJ, IEEE-CS, ACM.

Shuichiro Yamamoto received his B.S. in information engineering from Nagoya institute of Technology in 1977, and his M.S. in information engineering from Nagoya University in 1979, and his Doctor of Engineering degree from Nagoya University in 2000. He is a professor of Nagoya University. Previously, he was the first fellow of NTT Data research and development headquarters. He joined NTT in 1979. He moved NTT Data Corporation in 2002. And he moved Nagoya University in 2009.