Computer Systems Science & Engineering
DOI:10.32604/csse.2021.015157
Review

Tech Science Press

An Evaluation of Value-Oriented Review for Software Requirements Specification

Qiang Zhi[1,*] and Shuji Morisaki[2]

[1]Tokyo Institute of Technology, Tokyo, 152-8550, Japan
[2]Nagoya University, Nagoya, 464-8601, Japan
*Corresponding Author: Qiang Zhi. Email: zhiqiang0728@gmail.com
Received: 08 November 2020; Accepted: 11 January 2021

Abstract: A software requirements specification (SRS) is a detailed description of a software system to be developed. This paper proposes and evaluates a light-weight review approach called value-oriented review (VOR) to detect defects in SRS. This approach comprises setting core values based on SRS and detecting the defects disturbing the core values. To evaluate the effectiveness of the proposed approach, we conducted a controlled experiment to investigate whether reviewers could identify and record the core values based on SRS and find defects disturbing the core values. Results of the evaluation with 56 software engineers showed that 91% of the reviewers identified appropriate core values and 82% of the reviewers detected defects based on the identified core values. Furthermore, the average number of defects detected using the proposed approach was slightly smaller than that detected using perspective-based reading (PBR); however, PBR requires defining review scenarios before attempting to detect any defects. The results also demonstrated that the proposed approach helped reviewers detect the omission defects, which are more difficult to detect from SRS than defects because of ambiguity or incorrect requirements.

Keywords: Value-oriented review; perspective-based reading; scenario-based reading; lightweight review; software requirements specification

1 Introduction

Software review is an important part of the software development cycle that assists software engineers in detecting defects early [1], during the requirement and design phases. It is usually performed without executing programs. Detecting and correcting defects in the early stages of the software development life cycle reduces rework effort; therefore, software review methods and approaches that help reviewers efficiently detect defects are particularly valuable. Collaflello et al. [2] and Kusumoto et al. [3] proposed methods that minimize rework effort by measuring the impact of early defect detection and correction. In their experiment [3], subjects were divided into seven groups that worked to develop different software versions satisfying the same specifications. Experimental results showed that detecting and fixing defects during software testing took more effort than writing and reviewing the code.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Review guidelines help reviewers efficiently spot defects by establishing a detection procedure that does not treat every defect as equally important, but rather assigns priorities. Typical review methods include checklist-based reading (CBR) [4,5] and scenario-based reading (SBR) [6].

In CBR reviews, reviewers use a checklist as a guide. The checklist provides inspection objects but offers little guidance on how to perform the inspection. In SBR reviews, reviewers use scenarios as a guide; a scenario provides them a way to read software documents and detect defects. Many studies have investigated the effectiveness of SBR [7–12]. Perspective-based reading (PBR) [7,13–16] is a common SBR method that helps stakeholders such as software users, designers, and testers to focus on software requirements specification (SRS) problems. Guided reviews require additional effort and guides must be prepared beforehand; review scenarios should be tailored to the review goals [17,18].

In *ad hoc* reviews [19], reviewers do not receive any guidelines on how to conduct the review. The review can be as simple as reading the code and identifying defects sequentially. However, the efficiency and effectiveness depend on the reviewers' skill and capability. Many studies have compared reviews with and without guidelines (*ad hoc*). Results demonstrated that guided reviews outperformed reviews without guides [7–9,12]. Manual inspection (MI) [20] is an alternative when other inspection techniques are difficult to follow.

For SRS reviews, many software companies in Japan follow diametrically opposed approaches: they either spend considerable on conducting an exhaustive and comprehensive CBR or SBR review or do not spare any significant effort, and the project manager performs an *ad hoc* review. The value-oriented review (VOR) approach for requirement inspections presented in this paper is a lightweight approach to effectively and efficiently review requirements specifications without much preparation effort. In this approach, reviewers first specify the core values of the software, and then detect defects that disturb these core values. VOR does not require any complex guides; rather it requires specifying core values. We assume that VOR can be used in cases where review scenarios or guides do not exist or when defects related to software core features need to be found quickly and efficiently. Presumably, this is the first study that investigates the effectiveness of such an approach.

The remainder of this paper is structured as follows: Section 2 describes software review issues. Section 3 defines the VOR approach. Section 4 describes the setup of a controlled experiment to evaluate the VOR approach. Section 5 presents the experimental results. Section 6 discusses the results. Finally, Section 7 provided a summary of this paper.

2 Related Research

Gordijn et al. [21] investigated value-based requirements engineering for e-commerce system requirements and business process analysis. Lee et al. [22] pointed out that previous studies treated software review in a value-neutral setting and proposed a value-based review (VBR) process. Thew et al. [23] proposed a method for value-based requirements engineering to guide the elicitation of stakeholders' values, motivations, and potential emotional reactions. In our approach, software provides value for stakeholders' requirements and is largely defined by its features. Our approach differs from VBR in that the latter is based on CBR review methods. Furthermore, the "value" in our approach focuses on software features rather than economic values. The rest of this section presents concerns regarding the reading techniques reported by previous studies and describes the efforts required for preparing the review scenarios.

Denger et al. [24] developed a framework called TAQtIC to customize reviews for the target software by defining appropriate review scenarios for novice and expert developers. The main concerns in the novice developer scenarios were ensuring sufficient documentation, source code comments, error definition, exception handling in user interaction, and debug logging whereas those in the expert developer scenarios

were ensuring appropriate performance and time-memory trade-off, rounding error acceptability, and source code reusability. In Laitenberger et al. [25], defined PBR scenarios for UML documents. The PBR scenarios were constructed on the basis of perspectives of the tester, designer, maintainer, and requirements engineer. The tester's concern was to ensure sound operations in the object diagram. The designer's concern was to ensure that the operations met their responsibilities in terms of the object interactions. The maintainer's role was to ensure that the interactions among the objects in the collaboration diagram could be easily changed and maintained in the future. The requirements engineer's focus was on ensuring consistency among various operation descriptions in the analysis models. In Laitenberger et al. [26], conducted a controlled experiment with UML documents to compare and evaluate the effectiveness and efficiency of PBR and CBR. PBR included the perspectives of a tester, designer, and software programmer. The tester's role was to ensure software testability. The designer's role was to ensure the correctness and completeness of the design diagrams with respect to the analysis diagrams, and the software programmer's role was to ensure design consistency and completeness. Further, Laitenberger et al. [27] conducted a controlled experiment with code documents and 60 professional developers to compare and evaluate the effectiveness and efficiency of PBR and CBR. The PBR scenarios were constructed from the perspectives of the tester and code analyst. From the tester's perspective, the aim was to ensure that the implemented functionality was correct, and from the code analyst's perspective, the goal was to ensure that the expected functionality was implemented.

Defect-based reading (DBR) [6,8,28,29] aims to detect defects that are classified into specified types and provide guides for detection. Porter et al. [28] conducted a controlled experiment with 24 graduate students to compare and evaluate the effectiveness and efficiency of DBR, CBR, and *ad hoc* reading. The defect types were as follows: data type consistency, incorrect functionality, and ambiguities or missing functionality types. Porter et al. [6] utilized the requirement specification documents in [28]. Forty-eight graduate students were asked to compare and evaluate the effectiveness and efficiency of DBR, CBR, and *ad hoc* reading. In Porter et al. [8], sought to replicate the experiment of [28] asking professional developers to compare and evaluate the effectiveness and efficiency of DBR and *ad hoc* reading. The defect types of DBR scenarios were the same as those in Porter et al. [28].

In Thelin et al. [30] a similar approach to CBR was introduced, i.e., usage-based reading (UBR). In UBR, a predefined list of prioritized use-cases is used for design inspection. As inspection is a resource-intensive process [4], researchers and practitioners have sought to optimize the inspection by automating inspection process via tool-assisted techniques [31,32]. Recently, an IoT-based scenario description inspection method based on SBR named SCENARIOTCHECK was proposed [33]. Ali et al. [34] proposed an inspection process to enhance the quality of SRS using a mapping matrix and third-party inspection. Naveed Ali et al. [35] proposed a method for a situation in which the software teams are located in different parts of the world. Laat et al. [36] conducted an experiment to evaluate checklist quality; their results showed that checklist quality had a significant effect on SRS review. Because all proposed methods require considerable preparation and time, we consider a compromised approach, VOR, rather than an *ad hoc* one, when review scenarios or guides are unavailable.

3 Value-Oriented Review

3.1 Core Value

3.1.1 Definition

In this paper, we call core values the software features that are relevant to the stakeholders' requirements. In other words, core values can be considered important functional and non-functional requirements and can be determined by designers, developers, and users. Therefore, the core values defined in this paper can be considered system goals [37]. However, system goals can also be defined as the form decoupled from

implementations such as “high- profit in the business.” VOR excludes such goals that are decoupled from SRS reviews.

Core values can be defined using various abstraction levels. Similar to system goals, core values range from high-level strategic concerns to low-level technical requirements [37]. For example, for an ATM (Automatic Teller Machine), a high-level abstract core value is “providing safe transactions.” The corresponding low-level specific core values are “service for withdrawal, deposit, and transfer” and “protecting a banking account from spoofing.” Furthermore, a core value can be defined more specifically as “stopping service for 24 h when incorrect PIN codes are entered thrice consecutively”.

3.1.2 Defects Disturbing the Core Values

Defects disturbing the core values are inappropriate descriptions in SRS, that prevent software from providing the core values. Taking the example of ATM again, the defect disturbing the core value “stopping service for 24 h when incorrect PIN codes are entered thrice consecutively” can be the omission of the definition “the number of incorrect PIN codes is reset to zero when the correct PIN code is entered.” Nonfunctional requirements include defects that affect the core values as well as functional requirements. For an online communication system, although a description of user authentication exists in SRS, there is no description of the encryption of communications. An error that disturbs the core value “online communication is secure” should be an omission error.

3.2 VOR and the Procedure

We organize the basic principle of VOR into three Ws (Why · What · How) as follows:

Why: Purpose of system development

- Current Issues
- Goals (what it should be)
- Gaps between current issues and goal (issues to be resolved)

What: What are the functional and the nonfunctional requirements?

- Functional requirements: list of features to be implemented in the system
- Nonfunctional requirements: efficiency, processing speed, security, etc.

How: Specific usability and implementation methods (tasks similar to system design)

- Basic design
  - UI design
  - Functional design
  - Data design
- Detailed design
  - Class diagrams and sequence diagrams
  - System architecture
  - Technology for mounting each part

The review procedure is as follows.

1. ***Why***: Reviewers understand the background and purpose of software development and then read the SRS document.

2. ***What***: Reviewers identify and specify the software core values. The core-value identification method follows the goals specifying and reasoning methodology in Lamsweerde [37], which was introduced in our training seminar (Section 4), and then refines the top-level core values and extracts sublevel core values. As previously stated, unlike the goal definition [37], we exclude factors that are not directly related to software development and SRS based review. Reviewers may discuss core values with other software stakeholders (in our controlled experiment, discussions of the core value identification are not permitted).
3. ***How***: Reviewers detect defects that can disturb core value implementation.

In guided reviews, the review organizer first defines criteria for detecting defects. Then, the criteria are provided to reviewers as a guideline. Finally, the reviewer detects defects using the guideline. For VOR, reviewers should identify and specify core values before defect detection. In other words, PBR requires review scenarios to detect specific types of defects before defect detection. A review scenario for PBR generally comprises the following parts:

1. Introduction of role (perspective) and interest
2. How to extract information
3. Ask questions about the extracted information to help reviewers to detect defects

The effectiveness of PBR tends to depend more on the review scenarios, whereas that of VOR tends to depend more on the core value identification. Details about PBR and VOR are covered in our training seminars (See Section 4).

4 Experiment

4.1 Experiment Overview

The purpose of this experiment is to compare the number and type of defects detected using VOR and PBR. Moreover, this experiment evaluates whether reviewers can identify core values and detect defects that disturb core values based on the aforementioned procedure. Besides, we also study the effect of reviewers' experience and expertise on both methods.

The purpose of this experiment can be formulated in the form of the following research questions:

RQ1: Can reviewers identify the core values appropriately?

RQ2: Can reviewers detect defects disturbing the identified core values?

RQ3: How effective is VOR in defect detection compared with PBR?

RQ3.1: What is the number of defects detected by VOR as a percentage of that by PBR?

RQ3.2: Are the types of detected defects different between VOR and PBR?

RQ3.3: Does the experience of the reviewer affect the number of defects detected?

We propose two hypotheses:

Hypothesis 1: As there are several different perspectives and *elaborate guides for PBR, we assumed that PBR detects a larger number of defects*.

Hypothesis 2: Because identifying core values before detecting defects for VOR is necessary, the number of core value related defects detected by VOR and PBR might be similar.

We selected and planned an experiment in a training seminar for practitioners as an empirical investigation for the following reasons: First, a training seminar should meet both our and the participants' objectives. The purpose of the participants was to gain knowledge and skills. Our goal was

to investigate the effectiveness of VOR by statistical analysis with practitioners as subjects. Second, statistical analysis requires a certain number of subjects. We employed a training seminar to increase the number of subjects for providing meaningful results. Third, we judged that the benefit of the experiment in the training seminar exceeded the constraints and limitations that Basili et al. [7] noted. This training seminar is held once a year and must be designated as one day (six-and-a-half hours). The training comprises several lectures and three exercises. All participants attended the training seminar in the same lecture room. Most of them came from different companies.

4.2 Subjects and Material

The subjects were asked to review and document any defects found. They were also asked to answer a questionnaire on their software development experience.

Considering that the entire training takes six-and-a-half hours, the review material should not be too lengthy. Most participants can quickly understand the system behavior defined in the material without having a hands-on experience of developing such a system. We prepared three different SRS documents (Tab. 1).

Table 1: Experimental SRS documents and inspection methods

Document	Name	Words	Figures	Method
P	Used Car Sales Management System	577	2	*Ad hoc*
A	Internal Library Management System	1908	2	VOR/PBR
B	Internal Attendance Management System	1709	2	VOR/PBR

Document P is used for the pretest: it is a used car sales management system document. The groupings are based on the pretest scores. Documents A and B are used for the main experiment: Document A is an internal library management system document, and Document B is an internal attendance management system document for study meetings and invited talks. Documents P, A, and B are written in natural language (Japanese) with several figures. There are 577 words in Japanese and two figures in Document P, 1908 Japanese words and two figures in Document A, and 1709 Japanese words and two figures in Document B. We injected omission, ambiguity, incorrectness, and cosmetic defects in these documents. Omission, ambiguity, and incorrect types were defined as in Porter et al. [6]. Several software engineers advised regarding the experiment design. The designed defects were common in similar SRS documents. Documents A and B have the same number of defects. Both Documents A and B begin with a description of the background for developing the software.

4.3 Experiment Design

To reduce the impact of document content as well as the learning effect on experimental results, we randomly divided the subjects into four groups, i.e., X, Y, Z, and W, based on the pretest results. The experiment was designed as a crossover study [38]. The training seminar included three exercises. We assigned the pretest to Exercise 1, the first review to Exercise 2, and the second review to Exercise 3. In Exercise 1, the participants detected defects using *ad hoc* reading. The SRS document for the pretest is Document P for all subjects. After the lectures, the participants performed reviews using VOR and PBR in Exercises 2 and 3. The review techniques and review materials for Exercise 2 and 3 were different.

The details of the design are given in Tab. 2.

Table 2: Experimental design

Group	Exercise 2		Exercise 3	
	Technique	Document	Technique	Document
X	PBR	A	VOR	B
Y	VOR	A	PBR	B
Z	PBR	B	VOR	A
W	VOR	B	PBR	A

In this experiment, we prepared three review roles for PBR: the designer's, tester's, and user's perspectives, which are publicly available PBR scenarios [7]. The subjects were asked to select one out of the three perspectives based on their experience and preference.

4.4 Procedure

4.4.1 Pretest (Exercise 1)

1. The organizer of the training seminar distributed answer sheets among all participants.
2. The participants were asked to detect defects in Document P using ***ad hoc* reading** and record the defects in the defect list form.
3. The organizer collected and scored all the defect lists and then divided the participants into four groups (X, Y, Z, and W) according to the score.

4.4.2 Lecture

The lecturer introduced and explained VOR, PBR, and other techniques.

4.4.3 First Review (Exercise 2)

1. The organizer distributed answer sheets for Exercises 2 and 3 and review material for Exercise 2. The organizer distributed PBR scenarios for groups X and Z.
2. The participants were asked to detect defects in the distributed document (A or B, Tab. 2) using a specified review technique (VOR or PBR, in Tab. 2) within 40 min.

4.4.4 Second Review (Exercise 3)

1. The organizer distributed the review material for Exercise 3 as well as PBR scenarios for groups Y and W.
2. The participants were asked to detect defects in the distributed document (A or B, Tab. 2) using a specified review technique (VOR or PBR, in Tab. 2) within 40 min.

4.4.5 End of the Training Seminar

The subjects were required to submit the defect lists from Exercises 2 and 3. The overview of this controlled experiment is shown in Fig. 1.

4.5 Analysis Procedure

This section describes the analysis strategy for answering the research questions. Subsection 1 describes the value classification in the experiment. Subsection 2 presents the defect classification in the experiment. Finally, the comparison with PBR is explained in Subsection 3.

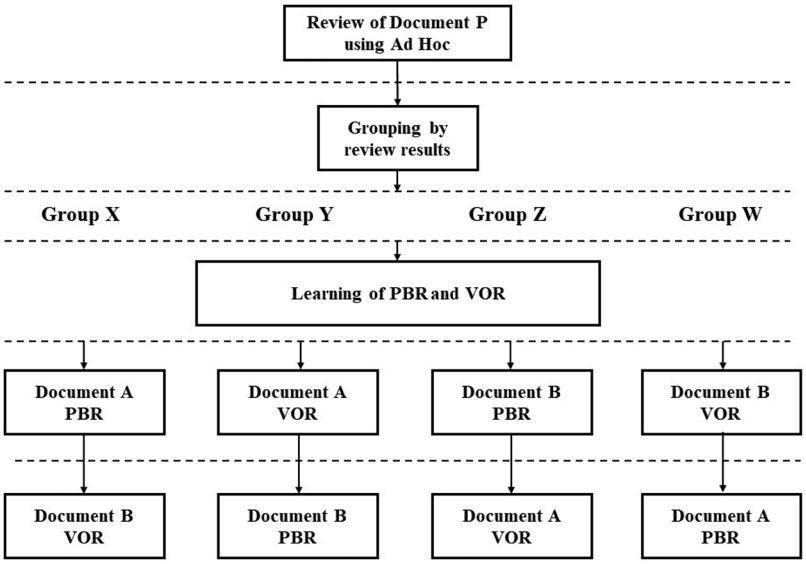

Figure 1: Overview of the controlled experiment

4.5.1 Core Values

Tab. 3 lists the top core values of Documents A and B. Tab. 4 gives the core values and examples of the corresponding disturbing defects. RQ1 would be evaluated based on whether the subjects identified the defined core values (top or lower level), and RQ2 would be evaluated based on whether the subjects who identified the core values detected the defects that disturbed the identified core values.

Table 3: Top core values of the target software

Software	Document	Core values
Book management system	A	Borrowing and returning books without the reception person
		Sharing books efficiently according to appropriate borrowing period
Attendance management system	B	Reducing the vacant seats by real-time cancelation and registration
		Reducing the burden on the organizers

4.5.2 Defect Classification

We classified the detected defects into five categories; Tab. 5 lists the categories and examples. A value-disturbing defect categorized as (I) refers to a defect that disturbs the realization of the defined core values (Tab. 4). A nonvalue disturbance defect categorized as (II) refers to a defect that does not disturb the core values in Tab. 4. A false-positive defect (III) is not a true defect caused by subjects' misunderstandings. A cosmetic error (IV) is a minor defect that does not cause serious defects, such as misspelling and layout improvement. A feature suggestion (V) recommends a desirable (but not required) feature.

4.5.3 Comparison with PBR

We compared VOR and PBR in terms of the number of detected defects. In PBR reviews, reviewers (perspectives) were expected to form a team. Therefore, we combined the subjects in a virtual team and

calculated the total average number of detected defects. This experimental method is consistent with the method in Basili et al. [7]. If a team detected the same defects, then the number of detected defects would be counted as one. We also compared the individual averages of the two methods. In PBR, the subjects should select a perspective before the review.

Table 4: Examples of top core value types and defects

	Type	Example of the disturbing defect
Aa	Borrowing and returning books without the reception person	The procedure of the specified borrow procedure is flawed.
Ab	Sharing books efficiently according to appropriate borrowing period	Delay notifications are incorrectly sent.
Ac	Other	GUI buttons on the screen should be aligned horizontally.
Ba	Reducing the burden of the organizers	Attendee list should be sent to the organizer before the study meetings and invited talks.
Bb	Reducing the vacant seats by real-time cancelation and registration	Waiting list feature in case of full seats should be implemented.
Bc	Other	The maximum length of the name of study meetings and invited talks should be defined and verified.

Table 5: Classification of defect types

Defect category		Description	Example
I	Core value disturbing defects	The defects disturb the realizations of core values	The waiting list feature is omitted.
II	Noncore value disturbing defects	The defects cannot directly disturb the realization of the core values indicated as Aa, Ab, Ba, and Bb.	The authentication and authorization server is not specified.
III	False-positive	Incorrect defects due to reviewers' misjudgment or unexplained opinion	Barcodes attached to each book should be replaced with RFID.
IV	Cosmetic error	The description error and format error, which will not cause serious problems	GUI buttons in the screen should be left-aligned not center-aligned.
V	Feature suggestions	Proposal for adding optional features	The meeting search feature should accept regular expressions.

Furthermore, we evaluated the impact of development experience on the defect detection results and which defect categories were easier to detect using VOR. From the defined perspectives, we evaluated the experimental results for RQ3.1 in terms of the number of defects detected by individuals and teams. For RQ3.2, we compared the percentage of the categorized defects between VOR and PBR. For RQ3.3, we evaluated the effect of the developer experience on the number of detected defects. In particular, we divided the subjects into senior and junior developers based on their experience.

5 Results

5.1 Experiment Subjects

Fifty-six software engineers were involved in this experiment. The subjects were divided into four groups, each comprising 14 subjects. Tab. 6 gives the distributions of the perspectives for PBR in each group.

Table 6: Number of subjects for each perspective in each group

Group	X	Y	Z	W
Designer	7	3	4	3
Tester	4	5	6	4
User	3	6	4	7

5.2 Identified Core Values (RQ1)

The high-level core values identified by the subjects using VOR are summarized in Tabs. 7 and 8. Most of the identified core values were the same as those defined in Tab. 3. Some of the identified core values were the subset of the high-level core values. Tabs. 9 and 10 give the percentage of the subjects who identified either the core values or the subset of the core values. Twenty-six out of 28 subjects identified the core values of Document A and 25 out of 28 subjects identified the core values of Document B. In total, 51 subjects (91.1%) identified the core values.

Table 7: Identified top-level core values in Document A (internal library management system)

Top-level core values	Partial subset of the core values
Aa: Borrowing and returning books without the reception person	Login screen for users with administrator privileges, users can identify themselves as an administrator or a user because administrators cannot borrow books
	The system should help users look for books with the location ID
Ab: Books can be effectively borrowed and returned	Notification regarding the return date and delay
	Book retrieve function
	Lending restriction on the number of books

Table 8: Identified top-level core values in Document B (Internal attendance management system)

Top-level core values	Partial subset of the core values
Ba: Reducing the burden on organizers	Meeting information such as starting time and meeting room can be changed after starting the registration
	Meeting notifications should be sent to the attendees before the meeting.
Bb: Reducing the vacant seats by real-time canceling and registration	The number of maximum attendees can be changed after starting the registration

Table 9: Percentage of the subjects who identified the core values in Document A

Core value	Number of subjects (Y)	Number of subjects (Z)	Percentage
Aa	10	9	67.9%
Ab	2	3	17.9%
Aa and Ab	2	0	7.1%

Table 10: Percentage of the subjects who identified the core values in Document B

Core value	Number of subjects (X)	Number of subjects (W)	Percentage
Ba	9	6	53.6%
Bb	2	4	21.4%
Ba and Bb	2	2	14.3%

5.3 Defects Disturbing the Core Values (RQ2)

Tab. 11 lists the number and percentage of subjects who detected defects that disturbed the core values (defect type-I in Tab. 5). For example, in Group Z, 12 subjects identified the core values, and nine out of 12 subjects detected the type-I defects. Forty-two out of 51 (82.4%) subjects detected type-I defects. Tab. 11 lists the number of subjects who detected type-I defects in each group. Tab. 12 gives the number of detected defects disturbing the core values in each group with VOR. One core value may correspond to several defects.

Table 11: Number of the subjects who detected the defects disturbing the identified core values in each group

Core value	Group X / Proportion	Group Y / Proportion	Group Z / Proportion	Group W / Proportion
Aa	–	7 / 70.0%	6 / 66.7%	–
Ab	–	2 / 100.0%	3 / 100.0%	–
Aa and Ab	–	2 / 100.0%	0 / 0.0%	–
Ba	7 / 77.8%	–	–	5 / 83.3%
Bb	2 / 100.0%	–	–	4 / 100.0%
Ba and Bb	2 / 100.0%	–	–	2 / 100.0%

Table 12: Number of detected defects disturbing the core values in each group

Core value	Group X	Group Y	Group Z	Group W
Aa	–	26	12	–
Ab	–	14	9	–
Ba	25	–	–	26
Bb	12	–	–	7

5.4 Comparison with PBR (RQ3)

5.4.1 Number of Defects (RQ3.1)

As described in Section 4, we measured the number of defects detected using PBR and VOR for a team comprising three subjects. In the PBR experiments, each subject actively chose the perspective they preferred (In fact, because of the size and number of subjects in the experiment, we were unable to evenly distribute the PBR perspectives.) According to the distributions of the perspectives for PBR in Tab. 6, we counted the data of eight PBR teams from groups X and W for Document A, and seven PBR teams from groups Y and Z for Document B. Notably, for a fair comparison of the data of virtual teams, we selected the VOR data from the same subjects who also conducted a review using PBR (although it is a crossover experiment). Tab. 13 lists the average number of detected defects for a team.

Table 13: Average number of detected defects for a team

Document	Method	Average
A	PBR	7.15
A	VOR	6.38
B	PBR	9.69
B	VOR	7.43

We also counted all subjects' data and performed a more in-depth analysis. Tab. 14 lists the defect statistics, including averages and confidence intervals. Tab. 14 reveals that the averages of the number of defects detected using PBR were larger in an individual comparison; however, the confidence intervals could overlap in almost all groups. The detected defects discussed here include types I, II, and V. Comparison of type-I (core-value disturbing defects) is provided in the Appendix.

Table 14: Statistics for the number of detected defects

Document	Group	Method	Average	Standard deviation	Lower endpoint	Upper endpoint
A	Y	VOR	4.36	2.38	2.93	5.78
	X	PBR	5.57	1.80	4.49	6.65
	Z	VOR	2.79	2.45	1.31	4.26
	W	PBR	4.36	2.44	2.90	5.82
B	W	VOR	3.71	1.79	2.64	4.79
	Z	PBR	4.50	2.82	2.81	6.19
	X	VOR	3.86	1.81	2.77	4.94
	Y	PBR	6.86	3.11	4.99	8.72

5.4.2 Number of Defects in Each Type (RQ3.2)

Figs. 2 and 3 show the proportion of detected defect types for VOR, PBR, and each role of PBR. In these figures, "PBR" represents the proportion of the average number of defects found from the designer's, tester's, and user's perspectives. The proportions of the omission defects detected using VOR were larger in both

documents (53% and 50% for VOR, 34% and 42% for PBR). The proportions of the feature suggestions between VOR and PBR were comparable (13% for both VOR and PBR in Document A, 22% and 18% for both VOR and PBR in Document B).

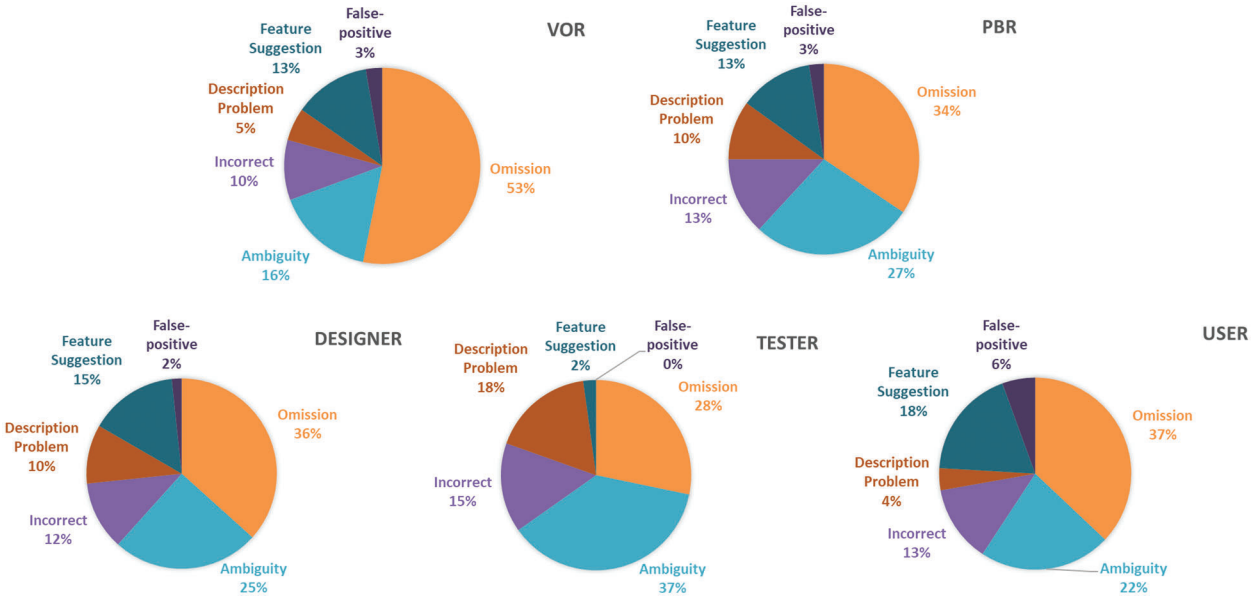

Figure 2: Proportion of the defects in six categories for Document A

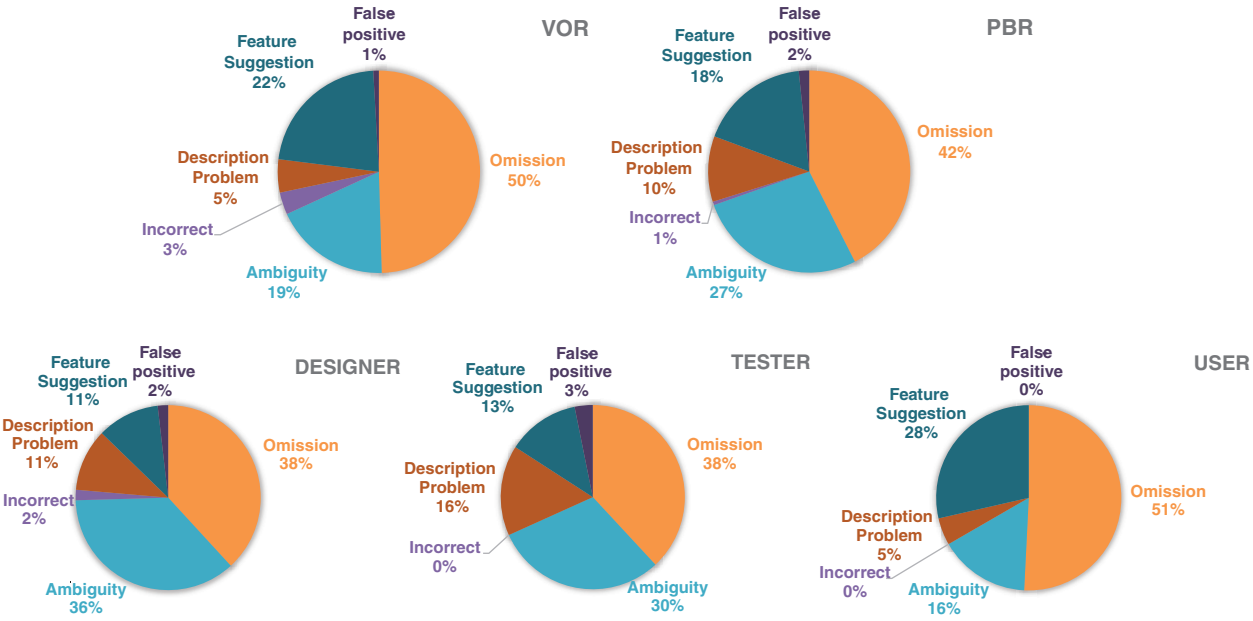

Figure 3: Proportion of the defects in six categories for Document B

Fig. 4 shows the average number of defects and the confidence intervals. The horizontal axes represent the defect types. The vertical axes represent the average number of defects. The number of omission defects was the greatest among all groups, and the number of omission defects detected by VOR was greater than

that detected by PBR. The average number of ambiguous defects detected by VOR was smaller than that detected by PBR.

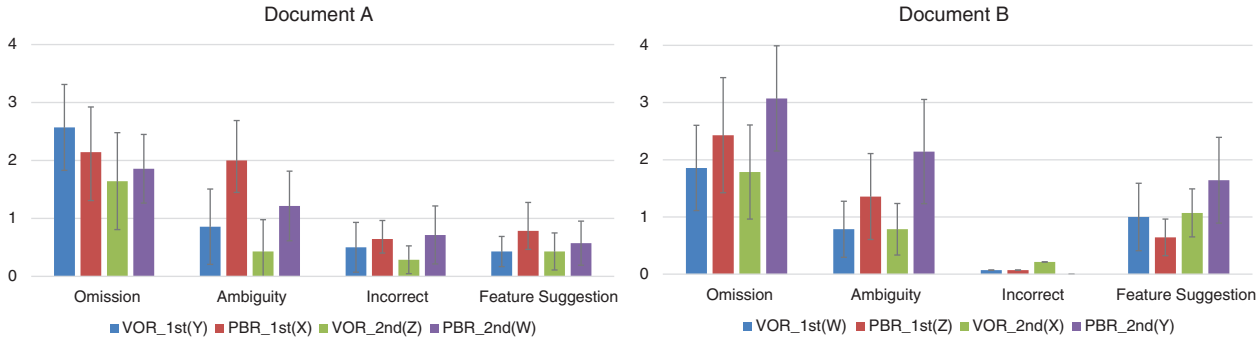

Figure 4: Defect comparison for the average number of detected defects and the confidence interval of each defect type for each document

5.4.3 Effect of Development Experience (RQ3.3)

We divided the subjects into junior and senior developers (>10 years of software development experience). All the subjects had more than three years of experience in software development with the average being ten years. Tab. 15 gives the average number of detected defects for the senior and junior developers using VOR and PBR, respectively. For both methods, the average number for the senior developers was greater.

Table 15: Average number of defects for senior and junior developers

	VOR	PBR
Senior developers	3.89	5.50
Junior developers	3.58	5.24

6 Discussion

6.1 Hypotheses

Hypothesis 1 can be confirmed from the experimental results. Nevertheless, the confidence intervals can overlap in almost all groups. For Hypothesis 2, the total number of type I defects detected using VOR is also smaller than PBR; however, the difference is minimal (See Appendix).

6.2 RQ1: Can Reviewers Identify the Core Values?

As seen in Tabs. 9 and 10, the answer to RQ1 is that over 90% of the subjects identified the core values or the subset of core values. The results indicated that the core values could be identified in different abstraction levels. We believe that exchanging the identified core values among the reviewers and stakeholders before defect detection activities can reduce the variation of identified core values. Moreover, based on the setup described in Section 3, modeling the core values by specifying and reasoning procedures [37] can reduce the variation.

6.3 RQ2: Can Reviewers Detect Defects Disturbing the Identified Core Values?

The answer to RQ2 is that 26 out of 28 subjects identified core values without relying on guides for Document A, and 20 out of 26 subjects detected the defects disturbing the identified core values. For Document B, 25 out of 28 subjects could identify core values without relying on guides, and 22 out of 25 subjects detected the defects disturbing the identified core values. In total, 91% of the reviewers identified appropriate core values and 82% detected the corresponding defects. In this experiment, subjects who could not identify the core values were also considered to be unable to detect the corresponding defects.

6.4 RQ3.1: What is the Number of Defects Detected by VOR as a Percentage of That by PBR?

The results of VOR are inferior compared with those of PBR. Regarding group comparison, the answer for RQ3.1 is 89.2% and 76.7% for Documents A and B, respectively (Tab. 13). Regarding the comparison for each subject, the confidence intervals among almost all groups can overlap according to Tab. 14. In this experiment, PBR realized comprehensive defect detection by assigning different perspectives to reviewers, whereas VOR needed to specify core values before defect detection. Besides, the core value identification and defect detection were conducted within a limited time frame.

6.5 RQ3.2: Are the Types of Detected Defects Different Between VOR and PBR?

The answer to RQ 3.2 is that VOR and PBR detect similar types of defects. The differences between Documents A and B with respect to ambiguity, incorrect, cosmetic, feature suggestion, and false-positive types were minimal. Further, it is encouraging that the average number of omission-type defects detected by VOR was greater than that detected by PBR in this experiment. For VOR, the percentages of the omission-type defects were 53% and 50% for Documents A and B, respectively. The omission-type defects are difficult to detect than other types of defects because reviewers must imagine the behaviors of the software and identify the omitted description for the behaviors.

6.6 RQ3.3: Does the Experience of the Reviewer Affect the Number of Defects Detected?

The answer to RQ 3.3 is yes; however, the impact was small. As seen in Tab. 15, the average number of defects detected by senior developers was greater than that detected by junior developers for both VOR and PBR. The differences between senior and junior developers were 0.31 and 0.26 for VOR and PBR, respectively. Thus, VOR may require more development experience from reviewers because it does not provide guides.

6.7 Implications

VOR enables a fast and cost-effective review, especially when review scenarios or guides are unavailable. In addition, the number of detected defects is expected to be greater than that in this experiment, because the core values were not shared among the reviewers in this experiment. In practice, core values can be shared among stakeholders, users, and other developers to improve effectiveness. Moreover, because VOR is a lightweight review technique, it can be performed in the early stages of requirement definition. Besides, combining VOR and PBR may improve the review quality by first performing VOR in an early stage and then using PBR as a gate check.

6.8 Threats to Validity

In this study, we took software engineers as the subjects and conducted experiments to evaluate the VOR method. However, it is difficult to simply state whether VOR is effective as it depends on several factors such as cost and accuracy. The following issues need to be considered for validating the experiment: how a human-oriented experiment affects the effectiveness [38], whether the constituent elements of the

experiment can be guaranteed, and whether the results of the experiment can be generalized to the actual working environment.

6.8.1 Internal Validity

The second review may have been affected by learning effects [38] because the experiment was performed twice in a day. In the reading technique literature, this problem can be suppressed by crossover-design [26,39]. As shown in Fig. 1, our controlled experiment was designed using the same philosophy. Besides, as the subjects were software engineers with software development experience, that the learning effect was expected to be smaller than that in the subjects who were students or software engineers with little or no experience.

6.8.2 External Validity

Because of the small scale of the SRS documents used in the experiment, the experimental results may not be generalized. However, this experiment could be considered a part of the large-scale software system review experiment. In other words, decomposing a large-scale software system into smaller parts before reviewing is also realistic, and large-scale software is indeed often decomposed in accordance with certain criteria before being reviewed.

6.9 Limitations

Although VOR performed efficiently (especially for the omission-type defects) with less cost in our controlled experiment, it is important to study the criticality degree of failure to identify defects. Besides, we suggest using VOR when review scenarios or guides are not available or the defects related to software core features need to be found quickly and efficiently: this is because in many cases, more defects should be detected even if costs increase.

7 Conclusion

In this paper, we proposed and evaluated VOR, which does not require preparation of review guides. In VOR, reviewers need to identify the core values of the target software before detecting defects. To evaluate VOR, we conducted a controlled experiment involving 56 practitioners. The results of the controlled experiment showed that 91% of the subjects identified appropriate core values and 82% of the subjects detected the defects disturbing the identified core values. The number of defects detected by VOR was slightly less than that detected by PBR. Regarding the individual comparison, the confidence intervals were overlapped. Besides, the total number of type-I defects detected by the two methods is similar. Furthermore, the results indicated that VOR helps the subjects detect the omission-type defects, which are difficult to detect than the other defect types. The results also showed that both junior and senior developers found an equal number of defects using VOR and PBR. Additionally, two of the companies that participated in the training seminar have piloted VOR in place of *ad hoc* reading within their premises and rated it favorably in our follow-up questionnaire. In summary, VOR can be implemented when review scenarios or guides are unavailable or when the defects related to software core features need to be found quickly and efficiently.

Funding Statement: This study was supported by the 2020 TSUBAME project of Tokyo Institute of Technology (Grant No. 20D10597).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. E. Fagan, "Design and code inspections to reduce errors in program development," *IBM Systems Journal*, vol. 15, no. 3, pp. 182–211, 1976.

[2] J. S. Collofello and S. N. Woodfield, "Evaluating the effectiveness of reliability-assurance techniques," *Journal of Systems and Software*, vol. 9, no. 3, pp. 191–195, 1989.

[3] S. Kusumoto, K. I. Matsumoto, T. Kikuno and K. Torii, "A new metric for cost effectiveness of software reviews," *IEICE Transactions on Communications Electronics Information and Systems*, vol. 75, no. 5, pp. 674–680, 1992.

[4] T. Gilb and D. Graham, S. Finzi (Ed.), *Software Inspection.* 75 Arlington Street, Suite 300 Boston, MAUnited States: Addison-Wesley Longman Publishing Co., Inc., 1993.

[5] B. Brykczynski, "A survey of software inspection checklists," *ACM SIGSOFT Software Engineering Notes*, vol. 24, no. 1, pp. 82–89, 1999.

[6] A. A. Porter, L. Votta and V. R. Basili, "Comparing detection methods for software requirements inspections: A replicated experiment," *IEEE Transactions on Software Engineering*, vol. 21, no. 6, pp. 563–575, 1995.

[7] V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull *et al.,* "The empirical investigation of Perspective-Based Reading," *Empirical Software Engineering*, vol. 1, no. 2, pp. 133–164, 1996.

[8] A. Porter and L. Votta, "Comparing detection methods for software requirements inspections: A replication using professional subjects," *Empirical Software Engineering*, vol. 3, no. 4, pp. 355–379, 1998.

[9] F. Shull, I. Rus and V. Basili, "How perspective-based reading can improve requirements inspections," *Computer*, vol. 33, no. 7, pp. 73–79, 2000.

[10] T. Thelin, P. Runeson and B. Regnell, "Usage-based reading—an experiment to guide reviewers with use cases," *Information and Software Technology*, vol. 43, no. 15, pp. 925–938, 2001.

[11] T. Thelin, P. Runeson and C. Wohlin, "An experimental comparison of usage-based and checklist-based reading," *IEEE Transactions on Software Engineering*, vol. 29, no. 8, pp. 687–704, 2003.

[12] M. Ciolkowski, O. Laitenberger and S. Biffl, "Software reviews: The state of the practice," *IEEE Software*, vol. 20, no. 6, pp. 46–51, 2003.

[13] O. Laitenberger, "Cost-effective detection of software defects through perspective-based inspections," *Empirical Software Engineering*, vol. 6, no. 1, pp. 81–84, 2001.

[14] V. R. Basili and F. Shull, "Developing techniques for using software documents: A series of empirical studies," Ph.D. dissertation, University of Maryland, Maryland, 1998.

[15] O. Laitenberger, "Cost-effective detection of software defects through perspective-based inspections," *Empirical Software Engineering*, vol. 6, no. 1, pp. 81–84, 2001.

[16] M. Ciolkowski, "What do we know about perspective-based reading? An approach for quantitative aggregation in software engineering," in *Int. Sym. on Empirical Software Engineering and Measurement*, Lake Buena Vista, FL, 2009.

[17] S. Biffl and M. Halling, "Investigating the defect detection effectiveness and cost benefit of nominal inspection teams," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 385–397, 2003.

[18] J. C. Carver, F. Shull and I. Rus, "Finding and fixing problems early: A perspective-based approach to requirements and design inspections," *Journal of Defense Software Engineering*, vol. 19, pp. 25–28, 2006.

[19] W. E. Lewis and W. H. C. Bassetti, *Software Testing and Continuous Quality Improvement*, 3rd ed. Imprint of Warren, Gorham and Lamont 31 St. James Avenue Boston, MAUnited States: Auerbach Publications, 2008.

[20] S. A. Ebad, "Inspection reading techniques applied to software artifacts - a systematic review," *Computer Systems Science and Engineering*, vol. 3, pp. 213–226, 2017.

[21] J. Gordijn and J. M. Akkermans, "Value-based requirements engineering: Exploring innovative e-commerce ideas," *Requirements Engineering*, vol. 8, no. 2, pp. 114–134, 2003.

[22] K. Lee and B. Boehm, "Empirical results from an experiment on value-based review (VBR) processes," in *Int. Sym. on Empirical Software Engineering*, Australia: Noosa Heads Qld., 2005.

[23] S. Thew and A. Sutcliffe, "Value-based requirements engineering: Method and experience," *Requirements Engineering*, vol. 23, no. 4, pp. 443–464, 2018.

[24] C. Denger and F. Shull, "A practical approach for quality-driven inspections," *IEEE Software*, vol. 24, no. 2, pp. 79– 86, 2007.

[25] O. Laitenberger and C. Atkinson, "Generalizing perspective-based inspection to handle object-oriented development artifacts," in *Proc. of the 1999 Int. Conf. on Software Engineering*, Los Angeles, CA, USA, 1999.

[26] O. Laitenberger, C. Atkinson, M. Schlich and K. E. Emam, "An experimental comparison of reading techniques for defect detection in UML design documents," *Journal of Systems and Software*, vol. 53, no. 2, pp. 183–204, 2000.

[27] O. Laitenberger, K. E. Emam and T. G. Harbich, "An internally replicated quasi-experimental comparison of checklist and perspective based reading of code documents," *IEEE Transactions on Software Engineering*, vol. 27, no. 5, pp. 387–421, 2001.

[28] A. A. Porter and L. G. Votta, "An experiment to assess different defect detection methods for software requirements inspections," in *Proc. of 16th Int. Conf. on Software Engineering*, Sorrento, Italy, Italy, 1994.

[29] A. Aurum, H. Petersson and C. Wohlin, "State-of-the-art: software inspections after 25 years," *Software Testing, Verification & Reliability*, vol. 12, no. 3, pp. 133–154, 2002.

[30] T. Thelin, P. Runeson, C. Wohlin, T. Olsson and C. Andersson, "Evaluation of usage-based reading—conclusions after three experiments," *Empirical Software Engineering*, vol. 9, no. 1/2, pp. 77–110, 2004.

[31] T. Thelin and P. Andersson, "Tool support for usage-based reading," in *The IASTED Int. Conf. on Software Engineering*, Innsbruck, Austria, 2004.

[32] F. Wedyan, D. Alrmuny and J. M. Bieman, "The effectiveness of automated static analysis tools for fault detection and refactoring prediction," in *Int. Conf. on Software Testing Verification and Validation*, Denver, CO, 2009.

[33] d. s. BP and R. C. Motta, "An IoT-based scenario description inspection technique," in *Proc. of the XVIII Brazilian Sym. on Software Quality*, Fortaleza, 2019.

[34] S. W. Ali, Q. A. Ahmed and I. Shafi, "Process to enhance the quality of software requirement specification document," in *Int. Conf. on Engineering and Emerging Technologies*, Lahore, 2018.

[35] N. Ali and R. Lai, "A method of software requirements specification and validation for global software development," *Requirements Engineering*, vol. 22, no. 2, pp. 191–214, 2017.

[36] M. D. Laat and M. Daneva, "Empirical validation of a software requirements specification checklist," in *24th Int. Working Conf. on Requirements Engineering*, Utrecht, Netherlands, 2018.

[37] A. V. Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proc. IEEE Int. Sym. on Requirements Engineering*, Toronto, Ontario, Canada, 2001.

[38] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell *et al., Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[39] J. Miller, M. Wood and M. Roper, "Further experiences with scenarios and checklists," *Empirical Software Engineering*, vol. 3, no. 1, pp. 37–64, 1998.

Appendix

As VOR focuses on defects related to the core features of software, discussing the comparison with PBR regarding the type-I defect (core-value disturbing defects) is also necessary. Each subject may describe the defects in different ways, but as long as the detected defect is related to the corresponding core value, it is considered a valid one. Besides, as mentioned in Section 3, core values can be defined according to various abstraction levels. An abstract core value can be decomposed into more specific core values.

Regarding Document A, the subjects of groups Y and Z conducted the review using VOR, and those of groups X and W conducted the review using PBR. The total number of defects related to core values detected by VOR and PBR are 61(63) and 67, respectively. However, the number of subjects of VOR was 26 instead of 28 because we excluded the data for subjects who did not identify core values as previously mentioned. Interestingly, one of the two subjects who did not identify core values also found type-I defects. Regarding

Document B, the subjects of groups X and W conducted the review using VOR, and those of groups Y and Z conducted the review with PBR.

The details are given in Tab. A1; "()" represents the data for all the subjects with VOR (subjects who did not identify core values are also included).

It can be seen that the total number of type-I defects detected using VOR is also slightly smaller than that using PBR.

Table A1: Type-I defect comparison between VOR and PBR for Documents A and B

Core values	Number of defects detected by VOR	Number of defects detected by PBR
Aa	38 (39)	41
Ab	23 (24)	26
Ba	51 (53)	45
Bb	19	30