An Experimental Comparison of Checklist-Based Reading and Perspective-Based Reading for UML Design Document Inspection

Giedre Sabaliauskaite
Dept. of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka
Osaka 560-8531, Japan
E-mail: giedre@ics.es.osaka-u.ac.jp

Fumikazu Matsukawa
Dept. of Computer Science, Graduate School of Information Science and Technology
Osaka University
1-3 Machikaneyama, Toyonaka
Osaka 560-8531, Japan
E-mail: matukawa@ist.osaka-u.ac.jp

Shinji Kusumoto
Dept. of Computer Science, Graduate School of Information Science and Technology
Osaka University
1-3 Machikaneyama, Toyonaka
Osaka 560-8531, Japan
E-mail: kusumoto@ist.osaka-u.ac.jp

Katsuro Inoue
Dept. of Computer Science, Graduate School of Information Science and Technology
Osaka University
1-3 Machikaneyama, Toyonaka
Osaka 560-8531, Japan
E-mail: inoue@ist.osaka-u.ac.jp

Abstract

This paper describes an experimental comparison of two reading techniques, namely Checklist-based reading (CBR) and Perspective-based reading (PBR) for Object-Oriented (OO) design inspection. Software inspection is an effective approach to detect defects in the early stages of the software development process. However, inspections are usually applied for defect detection in software requirement documents or software code modules, and there is a significant lack of information how inspections should be applied to OO design documents.

The comparison was performed in a controlled experiment with 59 subject students. The results of individual data analysis indicate that a) defect detection effectiveness using both inspection techniques is similar (PBR:69%, CBR:70%); b) reviewers who use PBR spend less time on inspection than reviewers who use CBR; c) cost per defect of reviewers who use CBR is smaller. The results of 3-person virtual team analysis show that CBR technique is more effective than PBR technique.

1. Introduction

For more than twenty-five years software inspections have been considered an effective and efficient method for defect detection. Inspections have been extensively investigated through controlled experiments in university environment and industry case studies. However, in most cases software inspections have been used for defect detection in documents of conventional structured development process, such as functional requirement documents or code modules [1,13,15,20]. There is a significant lack of information about how inspections should be applied to Object-Oriented (subsequently denoted OO) artefacts, such as OO code and design diagrams, because inspections were developed when the structured development process was dominant. Since over the past decade OO development methods have replaced conventional structured methods, it is very important to adapt the existing inspection techniques and develop new ones for OO artefact inspection.

Different reading techniques, which provide guidelines how to examine software artefacts and identify defects, are used during inspection. The most popular are Ad hoc and Checklist-based reading (subsequently denoted CBR) [13] techniques. The Ad hoc reading technique does not provide any instructions for the inspector on how to proceed during defect detection activity. CBR [6] provides the inspector with a checklist, which consists of procedural guidelines and "yes/no" questions. The inspector has to answer those questions while reading the software document. One more approach is the Scenario-based reading. It

provides inspector with a scenario, describing how to proceed and what to look for during inspection [12,14]. Several variants of scenario-based reading have been proposed: Defect-based reading [15], Reading technique based on function points [5] and Perspective-based reading (subsequently denoted PBR) [1,12,13,14]. The Defect-based reading concentrates on specific defect classes. The functional point based reading technique focuses on specific function point elements. PBR focuses on the perspectives (points of view) of the users of software documents.

To the best of our knowledge, little work has been done until now in the area of inspection of OO design document, written in the notation of Unified Modelling Language [4] (subsequently denoted UML). One of the examples is a controlled experiment described in [12]. In this paper, the authors experimentally compared two reading techniques, PBR and CBR. The comparison was made in a controlled experiment with eighteen subjects and two software systems. The results of that experiment showed, that 3-person inspection teams, which used PBR, had 41% effectiveness improvement and 58% cost per defect improvement over CBR teams. Other examples are described in [18,21], where the authors propose a set of reading techniques, called Traceability-Based reading (subsequently denoted TBR). The main idea of TBR is tracing information between design documents to ensure consistency (Horizontal reading), and between design documents and requirements to ensure correctness and completeness (Vertical reading). In [18] the authors presented an initial empirical study that was run to assess the feasibility of these reading techniques. The authors of those studies came to the conclusion that OO design document inspections need to be further investigated.

It is necessary to conduct software inspection experiments in different environments, using different people, languages, documents, etc. in order to understand all the aspects of software inspections more completely. We conducted a controlled experiment in Osaka University to compare CBR and PBR techniques for the UML diagram inspection. The subjects of the experiment were 59 third-year Bachelor students, and the language used during experiment was Japanese. Individual results as well as team results were compared. In the individual result comparison, we compared defect detection effectiveness, time spent on inspection and cost per defect of subjects who used CBR and PBR inspection techniques. In the team result comparison, we compared the defect detection effectiveness of CBR and PBR 3-person virtual teams. Briefly, the results of the experiment indicate that the subjects who used PBR have spent less time on inspection compared to those who used CBR, but they have greater cost per defect. The individual defect detection effectiveness for both inspection techniques is similar (PBR:69%, CBR:70%). Comparing the team results, however, we learn that the 3-person virtual teams that used CBR have higher defect detection effectiveness than the 3-person virtual teams that use PBR inspection technique.

The structure of the paper is as follows. Section 2 gives an explanation of two reading techniques – CBR and PBR. In section 3 the planning of the experiment is described. Section 4 describes the experimental subjects and objects. Threats to validity are described in section 5, and the analysis of the data is presented in section 6. In section 7 the results are discussed, and conclusions are given in section 8.

2. Checklist-based reading and Perspective-based reading

This section describes two reading techniques, namely Checklist-based reading and Perspective-based reading, which we used during our experiment.

2.1. Checklist-based reading

CBR has been a commonly used technique in inspections since 1970's. Checklists are based on a set of specific questions that are intended to guide the inspector during inspection.

In a survey of software inspection [13] the authors discuss a list of weak points of CBR. First, the questions are often general and not sufficiently tailored to a particular development environment. Secondly, concrete instructions how to use the checklist are often missing, i.e. it is often unclear when and based on what information an inspector is to answer a particular checklist question. Finally, the questions of the checklist are often limited to detection of defects that belong to particular defect types (inspectors may not focus on defect types not previously detected and, therefore, may miss whole classes of defects).

We have developed the checklist for UML diagram inspection by ourselves, based on the structure presented by Chernak [6] taking into consideration the weak points of CBR discussed in [13]. It consists of two components: "Where to look" (a description where to search for defects) and "How to detect" (a list of questions that should help the inspector to detect defects). The checklist contained 20 questions, and this is in line with the recommendations of Gilb and Graham [9] that a checklist should not be longer than a page (approximately 25 items).

2.2. Perspective-based reading

The main idea of the PBR technique is that a software product should be inspected from the perspective of

different stakeholders [1,2,3,12,13,14]. The perspectives depend on the roles people have within the software development and the maintenance process. For examining a document from a particular perspective, PBR technique provides guidance for the inspector in the form of a PBR scenario on how to read and examine the document.

The PBR scenario consists of three major sections [12,14]: introduction (describes the quality requirements, which are most relevant to this perspective); instructions (describe what kind of documents to use, how to read, how to extract the necessary information) and questions (set of questions which inspector has to answer during the inspection). The main objective of the use of instructions for reading a document from different perspectives is to gain a better defect detection coverage of a software artifact.

Three scenarios were developed for experiment: User scenario, Designer scenario and Implementer scenario. Inspection consisted of several steps. Each of them included the following information: diagrams to inspect, tasks to carry out, and questions to answer.

3. Experiment planning

In this section, the planning of the inspection experiment is described. It includes experimental variables, hypotheses and design of the experiment.

3.1. Variables

Two types of variables are defined for the purpose of the experiment: independent variables and dependent variables.

The independent variables include reading techniques, virtual team size and composition, duration of experiment, experience of subjects, etc. In our experiment, we let only reading techniques change, while other independent variables were kept constant. Reading techniques (CBR and PBR) were independent variables in our experiment.

We measured two types of dependent variables – dependent variables for individual subjects and dependent variables for 3-person virtual teams, which are described in Table 1.

Dependent variables for individual subjects were the variables calculated for each subject, such as the number of defects found, time spent on inspection, defect detection effectiveness, cost per defect. In addition, we measured average values of number of detected defects, time spent on inspection and defect detection effectiveness for subjects who used CBR and those who used PBR inspection technique. Although students have found more defects than it was seeded into UML diagrams, we decided to evaluate only a number of seeded defects found, because additional defects detected by students were not actually defects.

Dependent variables for the 3-person virtual teams were number of defects found by three members of the virtual team, maximum time spent on inspection, average number of unique defects detected by the team members and average team defect detection effectiveness. The time spent on inspection by the 3-person virtual teams was calculated choosing the maximum time spent by the inspectors of the team, because we wanted to compare the maximum time necessary for inspection using PBR and CBR. Method of combining subjects into virtual teams and of grouping those teams into statistically independent groups are described in Section 6.2.

Table 1. Dependent variables

Type	Dependent variable
For individual subjects	Number of defects found by a subject (DEF)
	Time spent on inspection (TIME), in minutes
	Defect detection effectiveness (EFF), in percent, calculated using the formula: EFF = (DEF / Total number of seeded defects) * 100
	Cost per defect (COST), in minutes, calculated using the formula: COST = TIME / DEF
	Average number of defects found by the subjects
	Average time spent on inspection
	Average defect detection effectiveness (AV_EFF), in percent, calculated using the following formula: AV_EFF = (Average number of defects found by subjects / Total number of seeded defects) * 100
For 3-person virtual teams	Number of unique defects found by 3 members of a virtual team
	Maximum time spent on inspection in minutes, calculated by choosing the maximum time spent by inspectors of the 3-person team
	Average number of unique defects found by the team members (AV_TEAM_DEF)
	Average team defect detection effectiveness (AV_TEAM_EFF), in percent, calculated using the following formula: AV_TEAM_EFF = (AV_TEAM_DEF / Total number of seeded defects) * 100

3.2. Hypotheses

We stated two types of hypotheses before the experiment: the hypotheses for the individual inspectors and a hypothesis for 3-person virtual teams.

3.2.1. Hypotheses for the individual inspectors. We assumed (H_{01}) that the subjects who used PBR technique during inspection should spend less time on inspection than those who used CBR, because a PBR scenario

covers only the UML documents related to a corresponding perspective and reviewer does not need to examine UML documents not related to his perspective. However, subjects who used CBR were supposed to examine all UML documents, and they needed to spend more time on the inspection. In addition, we assumed (H_{02}) that subjects who used PBR should have higher cost per defect, because they need not only to answer the questions, but also to perform various tasks before answering the questions. We did not know how different could the defect detection effectiveness of both methods be, so we only assumed (H_{03}) that PBR defect detection effectiveness should be different from CBR defect detection effectiveness.

Based on those assumptions, the following null hypotheses were stated:

H_{01}: Subjects spend more time on inspection using PBR than using CBR;

H_{02}: Cost per defect of subjects who use PBR is lower than the cost per defect of subjects who use CBR;

H_{03}: There is no difference in defect detection effectiveness of subjects who use PBR inspection technique as compared to subjects who use CBR.

3.2.2. Hypothesis for the virtual teams. We assumed (H_{04}) that team defect detection effectiveness should be different for PBR and CBR 3-person virtual teams. The following null hypothesis was stated for PBR and CBR inspector virtual teams:

H_{04}: There is no difference in defect detection effectiveness of the 3-person virtual teams, which use PBR inspection technique as compared to the 3-person virtual teams, which use CBR inspection technique.

3.3. Experimental design

An experiment consists of a series of treatments. To get the most out of the experiment, it needs to be carefully planned and designed [19]. When designing an experiment, it is necessary to look at the hypotheses and to see which statistical analysis it will be necessary to perform to reject the null hypotheses. Since we wanted to compare two inspection techniques CBR and PBR against each other, we chose design type of "one factor with two treatments" [19]. This is a simple experiment design for comparing two treatment means. Subjects are randomly assigned to each treatment, and each subject uses only one treatment on one object.

The design our experiment is shown in Figure 1. The reading techniques PBR and CBR are the treatments in our experiment. Each student participated only in one treatment (used either PBR or CBR reading technique during experiment), and inspected one software system (either Seminar or Hospital).

	PBR			CBR
	User	**Designer**	**Implementer**	
Seminar system	7	6	6	11
Hospital system	7	6	6	10

Figure 1. Experimental design

4. Experimental subjects and objects

This section discusses the subjects and objects used in the experiment. Subjects refer to the reviewers, and objects refer to the software artefacts inspected. The description of defects and experiment operation is given as well.

4.1. Experimental subjects

Subjects were 59 participants in the 3rd year of the Software Development course of Osaka University. They have had previous classroom experience with the programming languages, Object-Oriented development, UML, software design activities and conventional software review.

The class was divided into two groups of 29 and 30 students, and each group included subjects with the same mix of abilities (based on marks from Program Design class). Each group then focused on inspection of one software system. Inside each group, subjects were divided into two subgroups, each of them focused on only one inspection technique (PBR or CBR).

After the experiment, we asked the students to fill in a feedback questionnaire. The aim of this questionnaire was to collect subjective information on the level of difficulty (easy, medium, difficult) in understanding the software systems and a checklist or a scenario which they were using during the inspection; experience in software inspections, opinion about usefulness of such experiments in practice. Most of the subjects had no previous experience in software inspections.

4.2. Experimental objects

UML diagrams (paper-documents) of two software systems (Seminar system and Hospital system) were used as inspection objects. The Seminar system was dealing with the activities such as arrangement of seminar schedules, seminar hall reservation, lecturer designation, audience subscription, report reception and grading, etc. The Hospital system included activities such as oral consultation, medical examination, treatment of the patients, prescription of the medicines, etc. The number of diagrams for each system is given in Table 2. The

size of Seminar system documentation was 24 pages, and the size of Hospital system documentation was 18 pages.

At the beginning of the project, we held a training session in order to improve student's understanding of the software systems used. Students received description of the requirements, Use-case diagram and a part of the Class diagram, and were asked to create Sequence and Component diagrams of those systems.

During the experiment, system requirements description and Use-case diagram were assumed to be defect-free. The rest of the diagrams might contain defects. At least three defects were inserted into each type of UML diagrams (Class, Activity, Sequence and Component).

Students who were using CBR needed to inspect all the diagrams of the corresponding system. However, students who used PBR technique during inspection were inspecting only documents relevant to a specific perspective. The assignment of UML documents to inspection perspectives is shown in Table 2 ("U" corresponds to User's perspective, "D" – to Designer's perspective, "I" – to Implementer's perspective in Table 2). The assignment was based on a UML diagram development process, which students were learning during Software Design course. The main steps of this process are: the first step – development of Use-case diagrams; the second one – describing system activities in Activity diagrams; the third step – defining static structure of the system in Class diagrams; the fourth one – modelling dynamic aspects of the system in Sequence diagrams; the fifth step – detailed description of object states in Statechart diagrams; the sixth step – development of the Component diagrams. The User's perspective in our experiment covered the second, and partially the third and the fourth steps of software development process; the Designer's perspective covered the third and the fourth steps; and the Implementer's perspective covered the sixth step, and partially the third and the fourth steps.

Table 2. Experimental objects

UML diagram	Number of diagrams		CBR	PBR scenarios		
	Seminar system	Hospital system		U	D	I
Class	1	1	✓	✓	✓	✓
Activity	8	7	✓	✓		
Sequence	12	7	✓	✓	✓	✓
Component	1	1	✓			✓

4.3. Defects

In [18,21] authors describe defect taxonomy for UML design diagrams that previously had been proven effective for requirement's defects [2]. This taxonomy classifies defects by identifying related sources of information, which are relevant for the software system being built. Authors defined five types of defect: *Omission* (one or more design diagrams that should contain some concept from the general requirements or from the requirements document do not contain a representation for that concept); *Incorrect Fact* (a design diagram contains a misrepresentation of a concept described in the general requirements or requirements document); *Inconsistency* (a representation of a concept in one design diagram disagrees with a representation of the same concept in either the same or another design diagram); *Ambiguity* (a representation of a concept in the design is unclear, and could cause a user of the document to misinterpret or misunderstand the meaning of the concept) and *Extraneous Information* (the design includes information that, while perhaps true, does not apply to this domain and should not be included in the design).

We summarized defect taxonomy proposed by [18,21] authors into three types of defects: syntactic, semantic and consistency defects. Syntactic defects include Omission and Extraneous Information defects, semantic defects include Incorrect Facts and Ambiguity defects, and consistency defects correspond to Inconsistency defects.

In total fifteen defects were inserted into the software documents: 3 into the Class diagrams, 4 into the Activity diagrams, 5 into the Sequence diagrams, and 3 into the Component diagrams.

4.4. Experiment operation

Experiment was conducted in academic environment during a Software Development course in December 2001. The language of experiment was Japanese. The following timetable was used to arrange the experiment:

Week 1: Training session to improve student's understanding of the systems. The class was divided into two groups of 29 and 30 students. One of the groups received Requirement's description, Use-case diagram and part of Class diagram of a Seminar system. The other group received the above-mentioned documents of a Hospital system. Students were asked to create Sequence and Component diagrams of each system.

Week 2: Explanations of the experiment activities and conduction of the inspection experiment. Two rooms were used, one for each inspection technique – PBR and CBR. Students were divided into two groups: 38 (for PBR technique) and 21 (for CBR technique). Before the experiment students listened to the explanations, which lasted approximately 20 minutes. After the explanations were given, experiment was conducted. Experiment consisted of 120-minute (excluding explanations) individual inspection task. Students were

inspecting the same software system they had analyzed during the training session.

Week 3: Feedback questionnaire to collect additional information from students. The results from the questionnaire showed that inspectors who used CBR and those who used PBR had similar level of difficulty to understand checklist and scenarios, however inspectors who used PBR had better understanding of software systems they inspected. Most of the students had no previous experience in software inspection experiments, and most of them stated that such experiments could be useful in practice.

5. Threats to validity

There are four groups of threats to the validity of the experiment results: internal validity, external validity, conclusion validity and construct validity [19].

Threats to internal validity are treats that can affect the independent variable with respect to causality, without the researcher's knowledge. In our experiment there are no threats to history, maturation or mortality, because subjects participated only in one treatment and it lasted no longer than 2.5 hours. There might have been some threat to selection, because experiment was a mandatory part of the course. To minimize it, we have randomly assigned the subjects into groups which used only one of the reading techniques. In addition, we checked the groups to be similar in aspect of the level of student's knowledge. The objects (UML diagrams), which we used, could also have influence to the internal validity – threat of instrumentation. We made sure for both software systems to be similar in size and complexity. There was no risk for subjects to lack motivation, because students were told that the grading of the course would depend on their performance during inspection.

External validity concerns the ability to generalize the experiment results to industry practice. The biggest threat to the external validity is that students were used during the experiment as subjects. However, students were in the end of their third year of studies in software engineering, close to their start working in the industry. There are more experiments reported in the literature, where students were successfully used as subjects [10,17,18]. The design documents were similar to those which are used in practice, but the size of systems in industry is usually larger. However we think, that the amount of documents which subject were required to inspect was appropriate.

Threats to conclusion validity are concerned with the issues that affect the ability to draw the correct conclusion about the relationship between dependent and independent variables. Threats with respect to the subjects are limited, since we used third year students who have had similar knowledge and background, therefore there was no threat to random heterogeneity of subjects.

Construct validity concerns the ability to generalize from the experiment results to the concept or theory behind the experiment. The subjects did not know what hypotheses were stated, and they did not know the expected result of the experiment, so those threats to validity are considered small.

It can be concluded that there were threats to internal and external validity, but they were not considered large in this experiment. To increase the reliability of the results, replications of this experiment should be done.

6. Data analysis

This section describes the data collected during experiment and the statistical tests, which were used during data analysis. The section consists of 2 subsections: individual and team data analysis. In individual data analysis subsection, we analyze data of the individual inspectors. In team data analysis subsection, we combine the inspectors into virtual teams and analyze team results.

6.1. Individual data analysis

Two types of data were collected during the experiment, time data and defect data. Time data showed how much time each subject spent during the inspection. The added defect data showed the number of defects, which were detected by the subject. We calculated cost per defect (average time spent to detect one defect) and defect detection effectiveness (percentage of seeded defects which were detected) for every subject using formulas described in Table 1.

We compared time spent on inspection, cost per defect and defect detection effectiveness of subjects who used CBR and those who used PBR technique during inspection. The box-plots of those variables are shown in Figure 2, Figure 3 and Figure 4, and the statistics are given in Table 3. In the Figures 2-4, the box-plots graphically show the central location and scatter/dispersion of the data. The line on the left shows parametric statistics: the diamond shows the mean and the confidence interval around the mean, the notched line shows the requested percentile range. The notched box shows non-parametric statistics: the median, lower and upper quartiles, and confidence interval around the median.

As we can see from Table 3 and Figures 2-4, inspectors who used PBR inspection technique spent on the average 18% (11 min) less time on inspection than inspectors who used CBR. Cost per defect of inspectors

Table 3. Statistics of time spent on inspection, cost per defect and effectiveness

Variables	Reading technique	Number of subjects	Mean	SD	SE	95% CI of Mean	Median	IQR	95% CI of Median
Time spent on inspection	CBR	21	62.9	11.7	2.5	57.6 to 68.2	61.5	15.7	54.3 to 70.0
	PBR	38	51.3	15.1	2.5	46.3 to 56.3	50.0	17.1	44.2 to 59.2
Cost per defect	CBR	21	6.2	1.6	0.4	5.4 to 6.9	6.1	2.7	4.9 to 7.6
	PBR	38	10.2	3.5	0.6	9.0 to 11.4	9.9	4.1	8.5 to 11.6
Effectiveness	CBR	21	70.2	11.5	2.5	64.9 to 75.4	73.3	20.0	60.0 to 80.0
	PBR	38	69.1	15.3	2.5	64.0 to 74.1	69.1	22.0	66.7 to 77.8

who used CBR is 39% lower (4 min/defect) than of inspectors who used PBR. Inspectors who used PBR and those who used CBR exhibited similar defect detection effectiveness (about 70%).

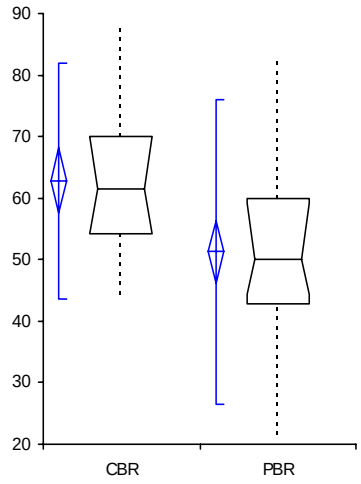

Figure 2. Time spent on inspection (minutes)

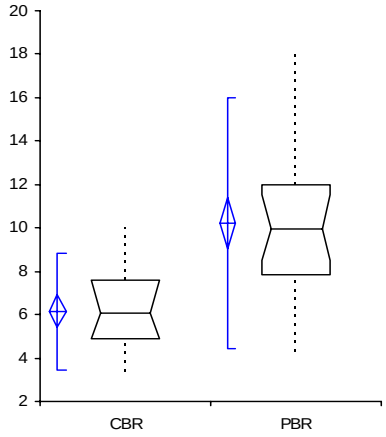

Figure 3. Cost per defect (minutes)

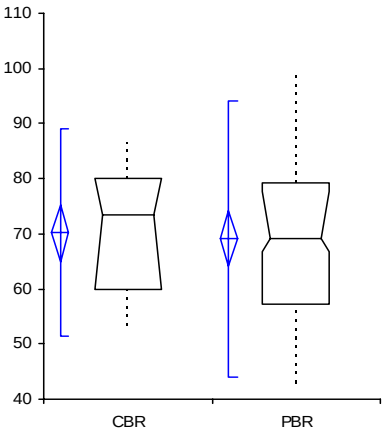

Figure 4. Defect detection effectiveness (%)

Parametric (Independent samples t-test) and non-parametric (Mann-Whitney) tests [11,19] were used to test the hypotheses for the individual inspectors.

The statistical results of testing hypotheses H_{01}, H_{02} and H_{03} are shown in Table 4 ("TIME" corresponds to time spent on inspection; "COST" corresponds to cost per defect; "EFF" corresponds to defect detection effectiveness in Table 4). The results of the statistical tests show that the hypotheses H_{01} and H_{02} can be rejected, but hypothesis H_{03} cannot be rejected.

Table 4. Statistics for t-test and Mann-Whitney test

Statistics	TIME (H_{01})	COST (H_{02})	EFF (H_{03})
t-test P value	0.0036	<0.0001	0.7769
t-test t value	3.04	4.97	0.28
Mann-Whitney test P value	0.0043	<0.0001	0.7145
Mann-Whitney test U value	212	102	376

In other words, it is statistically significant that

subjects spend more time on inspection using CBR inspection technique than using PBR inspection technique. In addition, it is statistically significant that cost per defect of subject who used PBR inspection technique is higher than the cost per defect of those who use CBR. However, there is no statistical significant difference in defect detection effectiveness of individual reviewers between CBR and PBR inspection techniques.

6.2. Team data analysis

Beside individual data analysis, team data analysis was performed. In this section, we describe the way to combine subjects into 3-person virtual teams and compare CBR and PBR team results.

6.2.1. Subject assignment to virtual teams. Beside individual result analysis, it is also important to evaluate team results for different reading techniques. Real team meetings [7,12,15,18] as well as virtual teams [1,3,20] are reported in literature. We decided to use virtual teams because we were more concerned with the range of team's defect coverage than with issues of interaction between members. We simulated team results by taking the union of the defects detected by the reviewers of the team.

The data of one reviewer using each of the three perspectives was included into PBR 3-person virtual teams, and any three reviewers were included into CBR 3-person virtual teams. The number of students who used CBR technique in Seminar system and Hospital system was different, therefore the number of unique teams for each system was also different: Hospital system (10 subjects) – 120 unique teams, Seminar system (11 subjects) – 165 unique teams formed. Both Seminar and Hospital systems were inspected using the same number of PBR inspectors, and the number of unique PBR teams was 252 for each system.

6.2.2. Grouping virtual teams into statistically independent groups. In the previous section, we described the way to combine subjects into teams. To avoid statistical dependence of the teams, we grouped them into statistically independent groups so that in each group the data of each subject was included only once.

To compare CBR and PBR 3-person virtual teams, six 3-person teams of inspectors who used PBR technique were combined into one PBR group, and three 3-person teams of inspectors who used CBR techniques were combined into one CBR group (Figure 5).

The number of CBR and PBR 3-person virtual team comparisons for Seminar system was calculated using the following formula (1):

$$\frac{{}_{11}C_3 \times {}_8C_3 \times {}_5C_3}{3!} \times \frac{{}_7P_6 \times 6! \times 6!}{6!} = 558835200000 \quad (1)$$

The number of CBR and PBR 3-person virtual team comparisons for Hospital system was calculated using the following formula (2):

$$\frac{{}_{10}C_3 \times {}_7C_3 \times {}_4C_3}{3!} \times \frac{{}_7P_6 \times 6! \times 6!}{6!} = 10160640000 \quad (2)$$

An example of comparison between PBR and CBR 3-person virtual team groups is shown in Figure 5 ("C" corresponds to subjects who used CBR technique during inspection; "U", "D" and "I" correspond to subjects who used PBR User's, Designer's or Implementer's perspectives of respectively).

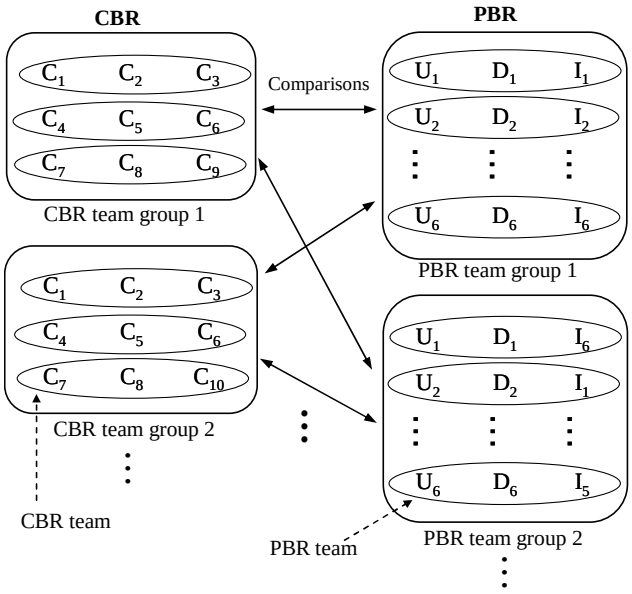

Figure 5. Comparison between CBR and PBR 3-person virtual team groups

6.2.3. Comparison of CBR and PBR virtual team results. PBR and CBR 3-person virtual team groups were compared with respect to defect detection effectiveness. The number of comparisons when either CBR or PBR was more effective, or both techniques were equally effective is shown in Table 5.

Table 5. Comparison of CBR and PBR 3-person virtual team groups

Software system	Team defects detection effectiveness		
	CBR>PBR	PBR>CBR	CBR=PBR
Seminar	54753326688	585743712	544449600
Hospital	10160482176	8064	149760

As we can see from Table 5, CBR teams have exhibited higher defect detection effectiveness that PBR teams for both Seminar and Hospital systems.

We used t-test with significance level of 2.5% to evaluate in which of comparisons either CBR or PBR inspection technique had significant difference with respect to team defect detection effectiveness. Out of comparisons, which exhibited significant difference in t-test, we counted the number of comparisons in which either CBR or PBR was more effective; or both inspection techniques were equally effective. We made two types of comparisons: including all types of defects and omitting syntactic defects. The results of comparisons are given in Table 6. As we can see from Table 6, in all comparisons between CBR and PBR 3-person virtual team groups, which showed significant difference in the t-test, CBR team groups exhibit higher defect detection effectiveness than PBR teams.

Table 6. Comparison of CBR and PBR 3-person virtual team groups after statistical tests

Comparison type	Software system	Defect detection effectiveness		
		CBR > PBR	PBR > CBR	CBR = PBR
All defects included	Seminar	7631557968	0	0
	Hospital	7233873848	0	0
Syntactic defect omitted	Seminar	6968160000	0	0
	Hospital	85122000	0	0

7. Discussion

In this section, an interpretation of the results is given. The following hypotheses show significant results:

H_{01} – Subjects spend more time on inspection using PBR than using CBR. (t-test P = 0.0036; Mann-Whitney test P = 0.0043)

H_{02} – Cost per defect of subjects who use PBR is lower than cost per defect of subjects who use CBR. (t-test P < 0.0001; Mann-Whitney test P < 0.0001)

H_{04}: There is no difference in defect detection effectiveness of 3-person virtual teams, which use PBR inspection technique, as compared to 3-person teams, which use CBR inspection technique (all CBR teams, which were significantly different from PBR teams using t-test with significance level of 2.5%, exhibited higher defect detection effectiveness than PBR teams).

The below hypothesis did not show significant results:

H_{03} – There is no difference in defect detection effectiveness of subjects who use PBR inspection technique as compared to subjects who use CBR. (t-test P = 0.7769; Mann-Whitney test P = 0.7145)

In other words, the results of the individual data analysis show that although subjects who use PBR technique spent 18% less time on inspection than subjects who use CBR technique, the cost per defect of PBR subjects was 39% higher. Although the individual data analysis did not show reasonable difference in defect detection effectiveness between PBR and CBR inspection techniques, virtual team data analysis showed that CBR teams were more effective than PBR teams.

The results of our experiment are in line with the results of several experiments of requirement and code inspections. In [20] authors collected data from the software inspection experiments reported in literature. In total, 21 data sets from the requirements phase and 10 data sets from code inspections were collected. The comparison of the effectiveness in inspection using different reading techniques (Ad hoc, CBR, PBR) showed that CBR was more effective than other reading techniques. In [7] authors reported on experiment of OO code inspection. The results of this experiment showed, that CBR emerged as the most effective approach.

8. Conclusion

The experiment presented in this paper is focused on comparison of two reading techniques, CBR and PBR for UML design document inspection. Experiment was run with 59 third year Bachelor students at the department of Informatics and Mathematical Science of Osaka University in December 2001. The language used during experiment was Japanese.

The results of the experiment indicate that time spent on inspection of subjects who use PBR is lower than of subjects who use CBR. However, the cost per defect of PBR subjects is higher as compared to CBR subjects. Defect detection effectiveness of 3-person virtual teams using CBR is greater than of those using PBR in our experiment.

Future research will be directed to further investigation of OO design document inspection.

9. References

[1] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, M.V. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading", *Empirical Software Engineering: An International Journal*, vol.1 , no. 2, 1996, pp. 133-164.

[2] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, M.V. Zelkowitz, "Lab Package for the Empirical Investigation of Perspective-Based Reading", Internet address: http:// www.cs.umd.edu/projects/SoftEng/ESEG/.

[3] S. Biffl, M. Halling, "Investigating the Influence of Inspector Capability Factors with Four Inspection Techniques on Inspection Performance", *Proc. of the Eighth IEEE Symposium on Software Metrics*, 2002, pp. 107-117.

[4] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley Longman, Inc., 1999.

[5] B. Cheng, R. Jeffery, "Comparing inspection strategies for

software requirements specifications", *Proc. of the 1996 Australian Software Engineering Conference*, 1996, pp. 203-211.
[6] Y. Chernak, "A Statistical Approach to the Inspection Checklist Formal Synthesis and Improvement", *IEEE Transactions on Software Engineering*, vol. 22, no. 12, 1996, pp. 866-874.
[7] A. Dunsmore, M. Roper, M. Wood, "Further Investigations into the Development and Evaluation of Reading Techniques for Object-Oriented Code Inspection", *Proc. of the Int. Conf. on Software Engineering*, 2002, pp. 47-57.
[8] M. Fagan, "Design and code inspections to reduce errors in program development", *IBM Systems Journal*, vol. 15, no. 3, 1976, pp. 182-211.
[9] T. Gilb, D. Graham, *Software inspection*, Addison-Wesley, 1993.
[10] M. Höst, B. Regnell, C. Wohlin, "Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment", *Empirical Software Engineering: An International Journal*, vol. 5, 2000, pp. 201-214.
[11] N. Juristo, A.M. Moreno, *Basics of Software Engineering Experimentation*, Kluwer Academic Publishers, 2001.
[12] O. Laitenberger, C. Atkinson, M. Schlich, K. El Emam, "An experimental comparison of reading techniques for defect detection in UML design documents", *The Journal of Systems and Software*, vol. 53, 2000, pp. 183-204.
[13] O. Laitenberger, J.M. DeBaud, "An encompassing life cycle centric survey of software inspection", *The Journal of Systems and Software*, vol. 50, no. 1, 2000, pp. 5-31.
[14] O. Laitenberger, C. Atkinson, "Generalizing Perspective-based Inspection to handle Object-Oriented Development Artifacts", *Proc. of the 21st Int. Conf. on Software Engineering*, 1999, pp. 494-503.
[15] A. Porter, L.G. Votta, V. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment", *IEEE Transactions on Software Engineering*, vol. 21, no. 6, 1995, pp. 563-575.
[16] B. Regnell, P. Runeson, T. Thelin, "Are the Perspectives Really Different? Further Experimentation on Scenario-Based Reading of Requirements", *Empirical Software Engineering: An International Journal*, vol. 5, no. 4, 2000, pp. 331-356.
[17] W. Tichy, "Hints for Reviewing Empirical Work in Software Engineering", *Empirical Software Engineering: An International Journal*, vol. 5., no. 4, 2000, pp. 309-312.
[18] G. Travassos, F. Shull, M. Fredericks, V. Basili, "Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality", *Proc. of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications*, 1999, pp. 47-56.
[19] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslen, *Experimentation in software engineering: an introduction*, Kluwer Academic Publishers, 2000.
[20] C. Wohlin, A. Aurum, H. Petersson, F. Shull, M. Ciolkowski, "Software Inspection Benchmarking – A Qualitative and Quantitative Comparative Opportunity", *Proc. of the Eighth IEEE Symposium on Software Metrics*, 2002, pp. 118-127.
[21] "Procedural Techniques for Perspective-Based Reading of Requirements and Object-Oriented Designs. Lab package of The Experimental Software Engineering Group (ESEG) of the University of Maryland", Internet address: http://www.cs.umd.edu /projects/SoftEng/ESEG.