Introducing Software Reading Techniques into Pattern Writer's Workshop: Checklists and Perspectives

Hironori Washizaki, Tian Xia, Yoshiaki Fukazawa
Global Software Engineering Laboratory, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo, Japan
washizaki@waseda.jp lmtc668800@moegi.waseda.jp fukazawa@waseda.jp

ABSTRACT
Pattern Writer's Workshop (WW) is a method to review, evaluate, and improve each other's pattern or pattern language papers under the guidance of a moderator. Although the WW has been well accepted at the pattern community and xPLoP conferences, there could be several problems such as poor moderations leading to "ad hoc" review meetings. To address the problems, we propose an approach for introducing two existing major software reading techniques, Checklist-Based Reading (CBR) and Perspective-Based Reading (PBR), to the WW targeting software patterns and pattern languages, and discuss potential benefits and drawbacks. Moreover we confirmed benefits and drawbacks of the approach by conducting an experiment.

Categories and Subject Descriptors
D.2.11 [**Software Architectures**]: Patterns

General Terms
Management, Documentation, Performance, Design, Experimentation, Human Factors

Keywords
Software pattern, Pattern language, Writer's workshop, Review, Checklist-based reading, Perspective-based reading

1. INTRODUCTION
Software patterns and pattern languages are usually reviewed and improved through shepherding process and Writer's Workshop (WW). The WW is a method to review, evaluate, and improve each other's pattern or pattern language papers under the guidance of a moderator [1][2].

Although the WW has been well accepted at the pattern community and xPLoP conferences to give authors review comments, there could be several problems such as poor moderations leading to "ad hoc" review meetings resulting in various problems such as few comments, superficial comments, and missing important concerns during WW.

To improve upon ad hoc review, there are various software reading techniques developed such as Checklist-Based Reading (CBR) and Perspective-Based Reading (PBR).

In this paper, we propose an approach for introducing software reading techniques, CBR and PBR, to the WW that targets software patterns and pattern languages. Moreover by conducting an experiment, the paper addresses three research questions in below.

RQ1. Does the WW with application of CBR and PBR contribute to more comments than the WW without CBR/PBR in a limited time?

RQ2. Does the WW with application of CBR and PBR contribute to concrete and profound comments (i.e. non-superficial ones) more than the WW without CBR/PBR?

RQ3. Does the WW with application of CBR (and PBR) contribute to reviewing and commenting on important concerns more than the WW without CBR/PBR?

Followings are the main contributions of this paper.

- A method for introducing CBR and PBR to WW.
- Drafts of a general checklist and specific checklists for several perspectives.
- Result of the experiment by conducting a WW with application of CBR and PBR revealing its effectiveness and limitation.

The rest of the paper is organized as follows. In Section 2, we introduce the WW with its typical problems and typical reading techniques. In Section 3, we propose an approach for introducing two reading techniques to the WW, and discuss potential benefits and drawbacks. In Section 4, we report a result of an experiment applying the approach to actual patterns and answer the research questions. Finally in Section 5 we conclude the paper and state some possible future works.

2. BACKGROUND AND PROBLEM STATEMENT
WW is a workshop for pattern and pattern language authors. Here we explain its traditional form and some limitations by showing a motivating example.

2.1 Writer's Workshop
Before a WW, all participants (i.e. authors) of the workshop group have to read others' patterns carefully and prepare comments. During the WW, papers are reviewed and discussed in several steps according to the following format. Notice that in the step (2) and (3), the author of the pattern under review should be "Fly on the wall," which means that he or she should only listen and record without joining in the discussion. Participants without the author make a circle like in Figure 1.

(1) The author of the pattern under review reads one or two important paragraphs in the pattern paper chosen by him/her.

(2) One of the participants summarizes the paper.

(3) Participants identify and praise strengths of the pattern.

(4) Participants identify the pattern's weaknesses that could be improved, and suggest possible improvements. If necessary, participants conduct a discussion based on the indication.

(5) The author gives words of thanks to all the participants and may ask questions of the participants to clarify their statements.

(6) Participants clap to thank the author for writing the paper.

(7) Participants submit the draft with comments to the author if they have.

According to the community of pattern such as xPLoP, the effectiveness of WW has been confirmed. Moreover there are several notes and patterns available for organizing successful WW in effective way, such as [1-3].

Figure 1. Circle of chairs for WW at MensorePLoP 2001

2.2 Problems in Traditional WW

Although the above-mentioned format of the WW is widely accepted by the pattern community, there are several problems as we experienced in previous xPLoP conferences. Figure 2 shows relationships among these problems P1-P3, their corresponding causes C1-C3, and proposed solutions S1-S2.

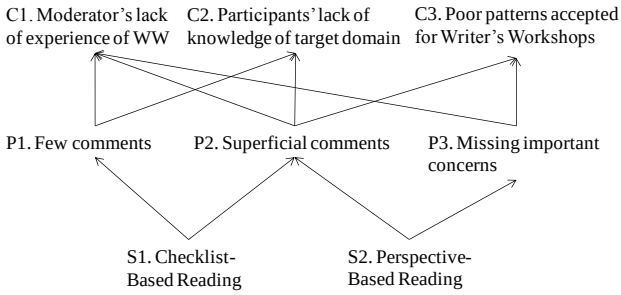

Figure 2. Relationships among causes, problems and solutions

The followings are explanations of these elements in the figure.

- C1. If a moderator lacks experiences of moderating WWs, the WW could be an "ad hoc" review meeting. Its typical symptom is a reading jumping around over many concerns or portions; it leads to having few comments (P1) and superficial comments (P2) since for all participants it is hard to follow the process and focus on import concern and/or portion in detail, while the time for WW is limited[1]. Moreover the moderator might not know what important concerns are and what are not while reviewing patterns so that the group might miss important concerns (P3) that should be considered.
- C2. If workshop participants including the moderator and other authors (i.e. reviewers) lack enough knowledge of the target domain of the pattern under review, it is hard to obtain many comments (P1) and comments about the pattern content in detail (P2).
- C3. Sometimes patterns do not have enough quality to be workshopped such as having improper form and missing forces. Although such poor patterns should be rejected or handled at the Writing Groups rather than WW, sometimes these poor pattern get a chance to be accepted for WW since xPLoP does not adopt rigorous review process but shepherding and voting process; if a corresponding shepherd with little experience and knowledge says "OK", then the pattern likely gets the acceptance. In such case, participants of WW spend a lot of time to mention general and superficial suggestions (P2) such as "improper form" rather than improvements of the pattern content in detail. It could lead to missing important concerns that should be considered (P3).
- P1. Although reviewers in WWs do not compete on number of review comments, authors of patterns to be workshopped wish to receive various and adequate amount of comments that are missed by the authors so that they can revise their patterns.
- P2. Review comments in WWs should be concrete and not superficial so that authors can easily understand and revise their patterns if necessary according to these comments.
- P3. There could be various concerns that could be incorporated in software patterns, such as general concerns mostly about pattern forms and readability, and, concerns specific to the pattern content such as specific quality characteristics.

Later in Section 3, we propose two techniques denoted as solutions S1-S2 to solve or mitigate these problems.

2.3 Software Reading Techniques

As we mentioned in above, the WW is a kind of review methods. Review is defined as a process or meeting during which a work product, or set of work products, is presented to project personnel, managers, users, customers, or other interested parties for comment or approval [4]. According to this definition, the WW can be seen as a meeting during which patterns or pattern languages are presented to other patterns' authors for comment.

In the area of software review such as design review and code inspection, there are various reading techniques developed to improve upon ad shoc review. A software reading technique is defined as a series of steps for the individual for a particular task [6-8]. Especially for the purpose of detecting defects in software artifacts, there are several well accepted reading techniques such as Checklist-Based Reading (CBR) and Perspective-Based Reading (PBR) [5-6]. Several empirical reports comparing CBR and PBR have been published [5]. CBR and PBR can be defined as follows.

- CBR is a reading technique, where the reviewers applying CBR use a list of statements or questions to be checked [6].

[1] Usually 1 hour for each paper.

- PBR assigns different perspectives to the reviewers to apply when inspecting a software artifact [6]. There are several particular perspectives for PBR, such as defect, functionality, usage [5] and stakeholders. PBR assumes that a specific focus performs better than a reviewer with the responsibility to detect all types of faults [6]. PBR also assumes that different perspectives can be designed so that their union yields full coverage of the inspected artifact.

Moreover sometimes these reading techniques are used together by preparing checklists for each perspective.

Software patterns and pattern languages are software artifacts, too. Moreover, by regarding defects as a kind of specific characteristics or concerns, there is a possibility to apply existing software reading techniques to software patterns or pattern languages in order to comment strengths and suggestions.

However to the best of our knowledge, it is unclear as to what kinds of software reading techniques are applicable and how are these techniques effective and efficient for WW.

3. READING TECHNIQUES FOR WRITER'S WORKSHOPS

Here we propose an approach for introducing two representative reading techniques, CBR and PBR, into the traditional WW.

3.1 Checklist-Based Reading

Method:

There could be two types of checklists to be used for the WW: a general checklist and some specific checklists corresponding to certain perspectives.

To create the general checklist, we first surveyed existing literatures on software pattern writing, shepherding, and criteria of software patterns. As a result, we identified six key literatures as sources to be used for creating the general checklist: original concept of patterns and pattern languages [9], essential characteristics of software patterns [10], essential elements of software patterns [11], rule of three [12], a pattern language for pattern writing [13] and a pattern language for shepherding [14].

Based on the sources, we propose a draft of the general checklist shown in Figure 3. In the figure 3, we classified 27 items into the following 11 categories.

- Being generative: According to the original concept of patterns and pattern languages, patterns are generative [9] (G1 in the figure 3). It should be applicable to any software patterns and software pattern languages; however it is somewhat hard to answer such conceptual question in software engineering. Therefore Winn and Calder identified nine items (G2-5, G14-16 and G21) premising that patterns are generative[2]. Among them, we considered four items (G1-I4) are particularly related to the nature of how patterns are generative.
- Domain and scope: A pattern, as a part of a larger pattern language, should focus on a certain scope with a certain domain.
- Structure: A pattern should use an appropriate form to clarify mandatory elements including a name, context, problem, forces, and a solution. Moreover a pattern often has optional elements including a resulting context, examples, and an acknowledgement.
- Problem and solution: A problem and a corresponding solution are the heart of a problem. The solution of a software pattern should capture software system hot spots.
- Forces: Forces should support problem in visible way. The forces of a software pattern should contain both functional issues and non-functional ones.
- Name and reference: A pattern should have appropriate name. A pattern should refer to other patterns explicitly.
- Known uses and validation: A pattern should be validated by use.
- Acknowledgement: In xPLoP, patterns are reviewed and improved by the shepherding process. Authors of pattern should appreciate for shepherds' efforts.
- Terminology and notation: A pattern should be comprehensive by using common terminology and figure notations.
- Pattern language: A pattern language, as a system of related patterns, should have a summary and a common running example.

Checklists could be used for both phases: preparation of the WW, and the WW in operation (especially steps (3) and (4) in Subsection 2.1). Regarding the former case, each participant can use a checklist while reading the pattern or pattern language to check typical strength and/or weakness, and prepare comments. Regarding the latter case, a moderator can use a checklist to ask participants what are typical strength and/or weakness of the pattern under workshop.

Benefit:

Benefit of CBR for WW is that participants or a moderator can easily and exhaustively review the pattern in terms of general concerns resulting in more comments (i.e. mitigation to the problem P1) even if the moderator has less experience knowledge or the participants have less knowledge.

Moreover at the same time some of items of the general checklist, can be a good starting point to discuss in detail so that it leads to some concrete comments rather than superficial ones (i.e. mitigation to the problem P2). Especially the items in the categories of "Being generative", "Problem and solution" and "Forces" are not simple Yes/No questions but profound ones requiring deep insights and discussions resulting in concrete and detail comments.

Drawback and countermeasure:

Drawbacks of CBR for WW and corresponding countermeasures are as follows:

- In CBR, participants and a moderator might be satisfied by just checking items in the given checklist superficially, especially if the given checklist has many items to be checked as shown in the figure 3; the participants and the moderator might overlook some important concerns specific to the give pattern or pattern language. Possible countermeasure could be that the moderator reads the pattern by using the checklist and summarizes answers for

[2] We merged "a pattern should be grounded a certain domain" [10] and "a pattern should be a part of a language" [10] into a single item G5 in the figure 3.

ID	Category	General item to be checked	Source
G1	Being generative	Does the pattern provide both a thing which is alive, and a process which will generate that thing?	[9]
G2	Being generative	Does the pattern imply an artifact?	[10]
G3	Being generative	Does the pattern bring many levels of abstraction?	[10]
G4	Being generative	Does the pattern leave inevitable mark on its application result?	[10]
G5	Domain and scope	Is the pattern grounded a certain domain and a part of a language?	[10][12]
G6	Domain and scope	Does the pattern have the right scope?	[13]
G7	Domain and scope	Is a target audience of the pattern clear?	[12]
G8	Structure	Does the pattern contain a pattern name, a context, a problem, a system of forces, and a solution?	[11][12]
G9	Structure	Does the pattern contain a resulting context, running examples and an acknowledgement if necessary?	[12][13]
G10	Structure	Does the form used fit the pattern content?	[13]
G11	Structure	Is the heart of the pattern (especially problem and solution) easy to access?	[12]
G12	Problem and solution	Does the problem and solution provide a big picture of the pattern?	[13]
G13	Problem and solution	Does the problem and solution match and fit together?	[13]
G14	Problem and solution	Is the solution strong in terms of capturing a big idea?	[10][13]
G15	Problem and solution	Does the pattern capture system hot spots?	[10]
G16	Forces	Does the pattern address both functional and nonfunctional issues?	[10]
G17	Forces	Does the forces explain what makes the problem difficult?	[13]
G18	Forces	Are forces highly visible regardless of the pattern form used?	[12]
G19	Name and reference	Does the name reveal the essence of its solution by noun phrase and meaningful meataphor?	[12]
G20	Name and reference	Does the pattern refer to other external patterns in understandable way?	[12]
G21	Known uses and validation	Is the pattern validated by use, preferably at least three times?	[10][12]
G22	Acknowledgement	Does the authors state their appreciation for their shepherd?	[13]
G23	Terminology and notation	Does the pattern use comprehensible terminology and figure notations?	[12]
G24	Terminology and notation	Is a glossary of terms provided?	[12]
G25	Pattern language	Is the pattern language summarized in its introduction?	[12]
G26	Pattern language	Is the summary of each pattern in terms of its problem and solution provided?	[12]
G27	Pattern language	Is the same running example used through the entire language?	[12]

Figure 3. Draft of the general checklist for software patterns and pattern languages

questions of the items before the WW, and show participants the summary at the beginning of the WW. Then the participants can select few items or other open concerns to be reviewed and discussed in detail during the WW.

- CBR might prevent participants from free and generative discussions leading to significant improvements or new patterns. Possible countermeasure is that the moderator could set a free discussion time in which checklists are not used.
- If the checklist to used is quite long, participants cannot check all items in the checklist since the time for WW is limited. Possible countermeasure is that a moderator selects or prioritizes some items in the checklist before WW.

3.2 Perspective-Based Reading

Method:

Existing perspectives for reading software materials are applicable to WW for software patterns and pattern languages, too. Such possible perspectives include quality characteristics, use cases and usage scenarios, and stakeholders.

- Quality characteristics: Any software pattern contributes to a software product, process, or resource (such as organizations). There are various quality models for software such as ISO/IEC 25010 quality model [15] for software products and IEEE Std 830-1998 Requirements Specification characteristics model [16] for software requirements. By using specific quality models, participants can exhaustively discuss how the pattern contributes to quality characteristics of the resulting software. For example in [15], there are five characteristics (effectiveness, efficiency, freedom from risk, satisfaction, and context coverage) as quality in use and eight as internal and external quality (functional suitability, compatibility, security, reliability, usability, performance efficiency, maintainability, and portability).
- Usecases and usage scenario: Participant could list up possible usecases and their scenarios of the resulting software, and discuss how the pattern contributes to that the resulting software works well for these usecases by using scenarios. Any pattern should contain at least one usecase (and corresponding scenarios).
- Stakeholders of resulting software: There could be various stakeholders involved in the resulting software, such as end-users, customers, requirement engineers, designers, programmers, testers, maintainers, quality engineers, process engineers, and managers. Different stakeholder view could reveal different concerns in the resulting software and how the target pattern contributes to these concerns. Figure 4 shows possible checklists for specific stakeholders of the resulting software based on existing checklists [17].
- Stakeholders of pattern: There are some stakeholders involved in the pattern itself: the original author of the pattern, authors of other existing or possible new patterns, pattern communities, and potential users of the pattern (such as software designers, programmers and maintainers for design patterns). For pattern users, most of items of the above-mentioned general checklist could be used to ensure

Stakeholder	ID	Considerations
End user	E1	Does the pattern contribute to satisfaction of needs and requirements in the resulting software?
	E2	Does the pattern contribute to revealing possible users and user behaviors of the resulting software?
	E3	Does the pattern contribute to ease of use of the resulting software?
Designer	D1	Does the pattern provide enough and consistent information for design?
	D2	Does the pattern contribute to adequate design complexity of the resulting software?
	D3	Does the pattern contribute to future extension and maintenance of the resulting software?
Tester	T1	Is the mechanism of the patttern solution reliable?
	T2	Is the realization of the pattern solution easy to test?
	T3	Does the pattern contribute to ease of testing of the resulting software?
	T4	Does the pattern provide enough information for testing the resulting software?
	T5	Does the pattern contribute to robustness of the resulting software for any input?

Figure 4. Draft of checklists for stakeholders of resulting software

that the pattern is useful for identifying problems, solving the problem, facilitating communications among users, and understanding software.

Perspectives could be used for both phases: preparation of the WW, and the WW in operation. For both cases, the moderator should define a list of perspectives, and assign different perspectives to participants as much as possible. To proceed PBR efficiently, it is better to prepare checklists corresponding to perspectives selected.

Benefit:

Benefit of PBR for WW is that by assigning different perspective to each participant, each participant can focus on the specific view and find strength and weakness in detail resulting in concrete comments rather than superficial ones (i.e. mitigation to the problem P3).

Moreover different perspectives can yield hopefully full coverage of concerns of the pattern (i.e. mitigation to the problem P2).

Drawback and countermeasure:

Drawbacks of PBR for WW and corresponding countermeasures are as follows:

- The effectiveness of PBR significantly depends on selection of appropriate perspectives to be used. A moderator has a responsibility to select appropriate perspectives corresponding to the target pattern and its domain. Otherwise, the participants and the moderator might overlook some important perspective specific to the given pattern, or use some less important perspectives. For example, the end-user's perspective might not work well for software architecture patterns since there could be a significant gap between the end-user's perspective (such as functionality) and architectural design. Possible countermeasure could be that the moderator could review checklists corresponding to perspectives and decide whether these perspectives fit the target before WW.
- PBR might prevent participants from free and generative discussions. Possible countermeasure could be that the moderator could set a free discussion time in which participants can review and discuss regardless of their assigned perspectives.
- If a checklist for a perspective is quite long, a participant assigned to the perspective cannot check all items in the checklist since the time for WW is limited. Possible countermeasure is that a moderator (or the participant) selects or prioritizes some items in the checklist before WW.

4. Experiment

To answer the above-mentioned three research questions RQ1-RQ3, we conducted an experiment for clarifying benefits and drawbacks of CBR/PBR for WW by introducing CBR and PBR into a WW at the focused group of Software patterns and Agile at IPSJ/SIGSE Winter Workshop 2015 on January 22-23[3].

4.1 Experiment Setting

As a target pattern for review, we choose "Enterprise Service Bus (ESB)" pattern from our paper [18] since most of participants know the concept of ESB somewhat. Moreover the paper was originally workshopped in PLoP 2011 by using the traditional WW form (hereafter "WW") having one hour and seven participants so that we can clarify the effectiveness of CBR and PBR compared with the traditional form although the paper has been slightly updated according to the result of PLoP 2011 WW.

There were six participants including two from us (i.e. the authors of this paper). We divided them into the following two groups:

- CBR group: Three participants used the general checklist shown in the figure 3.
- PBR group: Other three participants used three perspective-based checklists shown in the figure 4. Each participant took a different perspective: end-user, designer or tester.

Before the experiment, we only asked participants take a look at the target pattern. Both groups spent one hour for WW with application of CBR or PBR.

4.2 Experiment Result and Discussion

Figure 5 shows numbers of comments received at the original WW in PLoP 2011, the CBR group and the PBR group. The number of comments of CBR exceeded that of WW, while that of PBR is almost same as that of WW.

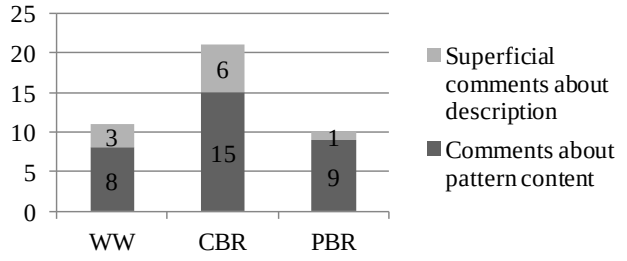

Figure 5. Numbers of comments received in WW, CBR and PBR

[3] http://wws.sigse.jp/2015/

In below we summarize how the comments were received and answer the research questions.

RQ1. Does the WW with application of CBR and PBR contribute to more comments than the WW without CBR/PBR in a limited time?

According to the figure 5, we confirmed that CBR contributed to more comments than WW. By checking the checklist given, the participants could indicate many comments in the limited time. On the hand, PBR contributed to less comments than WW since some of perspectives (especially the end-user's one) were hard to apply for the target pattern.

In summary, the answer is yes for CBR, and no for PBR. If authors wish to receive comments as much as possible, CBR could be a good solution.

RQ2. Does the WW with application of CBR and PBR contribute to concrete and profound comments (i.e. non-superficial ones) more than the WW without CBR/PBR?

In the original WW, there were three superficial comments such as "This article has a good structure" and "There is no exact example" although other comments were about the pattern content.

In the CBR group, most of the comments were about categories of "Being generative", "Domain and scope", "Structure", "Forces" and "Name and reference". Among them, there were some superficial comments simply saying "Yes" especially about categories of "Structure" and "Name and reference". In contrast, categories of "Being generative" and "Domain and scope" let the group discuss the pattern content in detail; these comments were not received at the WW without CBR/PBR.

In the PBR group, for all three perspectives, there were both positive and negative (improvement) comments. It seems that all of these comments except for one comment stating "This paper give several figures making designer easy to understand" were not superficial since each participant mentioned comments against the pattern content from the viewpoint of given perspective. The participants realized that for the pattern ESB, it was hard to use the end-user's perspective since there is a gap between systems high-level architecture posed by architecture patterns like ESB and end-users' concerns such as functionality and usability.

The answer is yes for CBR, and yes/no for PBR. CBR is superior to WW and WW with PBR since our general checklist contains profound items requiring deep insights and discussions resulting in concrete and detail comments. By choosing appropriate perspectives according to target patterns, the effectiveness of PBR regarding the quality of comments (i.e. profound comments or superficial ones) is thought to be improved.

RQ3. Does the WW with application of CBR (and PBR) contribute to reviewing and commenting on important concerns more than the WW without CBR/PBR?

CBR and PBR revealed comments, such as "There could be succeeding related patterns for implementing ESB in various different contexts" by CBR and "Testing for the whole system will be difficult" by PBR, which were not identified by the original WW without application of CBR/PBR.

On the other hand, there were some comments only appeared in WW without application of CBR/PBR, such as "Multiple ESB servers can be considered".

In summary it is hard to answer this research question by only the limited experiment. Perhaps authors (and facilitators) could specify important concerns according to their intentions such as validating the target pattern, revising the pattern, or discovering related patterns. According to such specification, WW participants could focus on important concerns by preparing and focusing on appropriate checklists and/or perspectives.

5. CONCLUSION

In this paper, we identified some possible problems that we observed in traditional writer's workshops (WWs). To address the problems, we proposed an approach for introducing two major existing software reading techniques, Checklist-Based Reading (CBR) and Perspective-Based Reading (PBR), to the WW targeting software patterns and pattern languages, and discussed potential benefits and drawbacks. Moreover we confirmed benefits and drawbacks of the approach by conducting an experiment.

In the experiment, we used three selected perspectives and checklists in CBR/PBR. In the future we will consider to refine and/or extend the checklists by referring to other literatures and conducting additional experiments. Moreover we will consider using other perspectives and comparing them in terms of usefulness and specificity to patterns and/or domains.

As another future work, we will consider the applicability of our approach to any non-software patterns; that could lead to reveal different perspectives. Moreover we will consider the applicability of our approach to other pattern activities such as pattern writing and shepherding.

6. ACKNOWLEDGMENTS

Our thanks go to shepherd Y C Cheng for very careful review that improve this paper. We thank to participants (including Jonatan Fernandez, Hideyuki Kanuka, Kosaku Kimura and Eiichi Hanyuda) of the focused group of Software patterns and Agile at IPSJ/SIGSE Winter Workshop 2015 for conducting the experiment. Also we thank AsianPLoP'15 Writers' Workshop participants for their valuable comments.

7. REFERENCES

[1] Hillside Group, "How to Hold a Writer's Workshop," and "Suggestions for a Successful Writer's Workshop," http://hillside.net/component/content/article/65howtorunplop/235howtoholdawritersworkshop

[2] Richard P. Gabriel, "Writers' Workshops & The Work of Making Things," Addison Wesley, 2002.

[3] James O. Coplien, "A Pattern Language for Writers' Workshops," in Pattern Languages of Program Design 4, Addison-Wesley, 2000

[4] IEEE Std 610.12-1990 (R2002), "IEEE Standard Glossary of Software Engineering Terminology," 2002.

[5] Keun Lee, "Development and Evaluation of Value-Based Review (VBR) Methods," VDM Verlag, 2008.

[6] Thomas Thelin, Per Runeson and Claes Wohlin, "An Experimental Comparison of Usage-Based and Checklist-Based Reading," IEEE Transactions on Software Engineering, Vol.29, No.8, pp.687-704, 2003.

[7] Forrest Shull, "Software Reading Techniques," Encyclopedia of Software Engineering, John Wiley & Sons, 2002.

[8] Forrest Shull, Jeffrey Carver, Guilherme H. Travassos, Jose Carlos Maldonado, Reidar Conradi, Victor R. Basili, "Replicated studies: building a body of knowledge about software reading techniques," Lecture notes on empirical software engineering, pp.39-84, 2003.

[9] Christopher Alexander, "The Timeless Way of Building," Oxford University Press, New York, 1979.

[10] Tiffany Winn and Paul Calder, "Is This a Pattern?," IEEE Software, Vol.19, No.1, pp.59-66, 2002.

[11] Deepak Alur, John Crupi, Dan Malks, "Core J2EE Patterns: Best Practices and Design Strategies," Pearson Education, 2001.

[12] Will Tracz, "RMISE Workshop on Software Reuse Meeting Summary," Software Reuse: Emerging Technology, IEEE CS, 1988.

[13] Gerard Meszaros and Jim Doble, "A pattern language for pattern writing," Pattern languages of program design 3, pp.529-574, Addison-Wesley, 1997.

[14] Neil B. Harrison, "The Language of Shepherding: A Pattern Language for Shepherds and Sheep," Pattern Languages of Program Design 5, Addison-Wesley, 2006.

[15] ISO/IEC 25010: 2010 Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2010.

[16] IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, 1998.

[17] Hiroyuki Okamoto, et al., "An Experimental Evaluation of Individual Review Methods Focusing on Software Requirements Specifications Characteristics," SQiP Study Group Report, 2004. http://www.juse-sqip.jp/workshop/seika/2004/6/6_report.pdf

[18] Eduardo B. Fernandez, Nobukazu Yoshioka, and Hironori Washizaki, "Two patterns for distributed systems: Enterprise Service Bus (ESB) and Distributed Publish/Subscribe," 18th Conference on Pattern Languages of Programs (PLoP), 2011.