An Experimental Comparison of Usage-Based and Checklist-Based Reading

Thomas Thelin and Per Runeson
Dept. of Communication Systems,
Lund University
Box 118, SE-221 00 LUND, Sweden
{thomas.thelin, per.runeson}@telecom.lth.se

Claes Wohlin
Dept. of Software Eng. and Computer Science
Blekinge Institute of Technology
Box 520, SE-372 25 Ronneby, Sweden
claes.wohlin@bth.se

Abstract

Software quality can be defined as the customer's perception of how a system works. Inspection is a method to control the quality throughout the development cycle. Reading techniques applied to inspections help reviewers to stay focused on the important parts of an artefact when inspecting. However, many reading techniques focus on finding as many faults as possible, regardless of their importance. Usage-based reading helps reviewers to focus on the most important part of an software artefact from a user's point of view. This paper is an extended abstract of a technical report describing an experiment, which compares usage-based and checklist-based reading. The results show that reviewers applying usage-based reading are more efficient and effective in detecting the most critical faults from a user's point of view than reviewers using checklist-based reading. Usage-based reading may be preferable to use for software organisations utilising or will start utilising use cases in their software development.

1. Introduction

Software inspections have since its inception [5] 25 years ago spawned quite some interest both from the research community and industrial practice. The research includes changes to the inspection process, e.g. [17][2][13][9], support to the process, e.g. [1][4], and empirical studies, e.g. [19][22]. The suggested improvements include active design reviews [17], two-person inspection teams [2], n-fold inspections [13], phased-inspections [9], perspective-based reading (PBR) [1] and the use of capture-recapture techniques to estimate the remaining number of faults after an inspection [4]. Industry has studied the benefits of conducting software inspections [25]. Moreover, software inspections have been popularised by books on the subject [6][3].

In parallel with the development of software inspections, software engineering as such has evolved. Two directions of evolvement are of particular importance in the context of this paper, namely usage-based testing [12][15] and the introduction of use cases in object-orientation [7]. One important common denominator of these two techniques that have emerged is the focus on usage. Based on this, a method for usage-based inspection was proposed by Olofsson and Wennberg [16]. The basic idea was to let the expected usage govern the inspection. The motivation behind the method was that faults that affect the user of the software the most are crucial to find, and hence an inspection method putting the user in focus was needed.

The initial idea has since been refined and further studied and some results have been presented by Regnell et al. [20] and Thelin et al. [24]. These two studies have refined the ideas and formulated the approach as a new reading technique denoted Usage-Based Reading (UBR). The studies have primarily focused on improving UBR as such. The objective here is to compare and hence evaluate how good the usage-based reading is in comparison with other methods. The paper presents a controlled experiment where usage-based reading is compared with checklist-based reading (CBR). The results are promising since the study shows that UBR is significantly better than CBR in terms of both effectiveness and efficiency in finding the faults that affect the user the most.

The paper is structured as follows. In Section 2, the background and principles of UBR are presented. In the following sections, the experiment is presented; experiment preparation in Section 3, experiment planning in Section 4 and experiment operation in Section 5. In Section 6, the analysis of experiment data is presented and in Section 7, the results are discussed. Finally, summary and conclusions are presented in Section 8.

2. Usage-Based Reading

Many reading techniques focus on finding as many faults as possible, regardless of their importance. The inspection effectiveness is often measured in terms of number of faults found, without taking into account that some faults in the inspected object are likely to affect the final system quality more than others do.

The principal idea behind Usage-Based Reading (UBR) is to focus the reading effort on detecting the most critical faults in the inspected object. Hence, faults are *not* assumed to be of equal importance, and the UBR method is aimed at finding the faults that have the most negative impact on the users' perception of system quality. The UBR method focuses the reading effort guided by a prioritized, requirements-level use case model.

In order to specify the users' perception of system quality, use cases are prioritized. The order of the use cases reflects what a user or a group of users thinks is most important in the system to be developed. The prioritization can be made by, for example, pair-wise comparisons according to the Analytical Hierarchy Process [8].

UBR utilises a set of use cases as a vehicle for focusing the inspection effort, much the same way as a set of test cases focuses the testing effort. The use cases tell the reviewers how to inspect a design or code document in a similar manner as the test cases tell the testers how to test the system.

Inspection of a design document using UBR is performed in the following basic steps:

- **Preparation** – Glance through the design document to be inspected, the use cases utilised to guide the reading and the requirements document, which is the reference to which the design is compared.
- **Inspection** – Start with the first use case. Trace the use case through the design document and use the requirements document as a reference. Identify anomalies in the design document and report the faults found. Repeat the inspection procedure using the next use case, until all use cases are covered, or until a time limit is reached.

Two variants of the UBR method are defined, *rank-based reading* and *time-controlled reading.* The former prioritizes the use cases with respect to the importance from a user's perspective. A reviewer using rank-based reading follows the use cases in the order they appear in the ranked use case document. Time-controlled reading adds a time budget to each use case in order to force a reviewer to utilise a specific use case the specified time. Time budgets are applied to each use case and are normally longer for use cases given a high rank and less for use cases given a lower rank. A detailed description of UBR is given by Thelin et al. [24].

In this investigation, rank-based reading is used. To investigate the effects of using UBR, the following research questions are addressed:

- **RQ1** – Is UBR more effective than CBR in finding the most critical faults?
- **RQ2** – Is UBR more efficient than CBR in terms of total number of critical faults found per hour?
- **RQ3** – Are different faults detected when using UBR and CBR?
- **RQ4** – Is UBR more effective and efficient than CBR considering the performance of an inspection team?

UBR is related to perspective-based reading (PBR) [22] in the sense that both reading techniques utilise the user perspective. In PBR, different perspectives are used to produce artefacts during inspection. The reviewers applying the user perspective develop use cases based on the inspected artefact and thereby find faults. In UBR, the use cases are used as a guide through the inspected artefact. The differences are hence that reviewers applying UBR *utilises* existing use cases while reviewers applying PBR actively *develops* use cases. The goal of UBR is to improve efficiency and effectiveness by directing the inspection effort to the most important use cases from a user's perspective, while PBR has the goal of improving efficiency by minimising the overlap among the faults that the reviewers find. The latter is, however, not always achieved [20]. There is no contradiction between the techniques, but they aim at fulfilling different goals.

3. Experiment Preparation

This section describes the preparation needed to conduct the experiment and the subjects acting in the experiment. Since the experiment is based on an experimental package developed at Lund University, most of the software artefacts are already described in a previous study. In this section, only a brief overview of the package is provided. For a detailed description of the artefacts included in the experimental package and how they were developed, see Thelin et al. [24].

3.1. Reviewers

The students participating as reviewers in the study were fourth-year Software Engineering Master students at Blekinge Technical Institute of Technology in Sweden. Many of the students have extensive experience from software development. As part of their bachelor degree, they have obtained extensive practical training in software development. Among other things, they have participated in a one semester project including 15 students. The customer for these projects are normally people in industry, and hence the students have participated in projects close to an industrial situation with changing requirements and time pressure. Several of the master students also work in industry in parallel with their studies. This means that the students are rather experienced and to some extent comparable to fresh software engineers in industry.

The experiment was a mandatory part of a course in verification and validation. The course included lectures and assignments both related to verification and validation of software products and evaluation of software processes. The latter means that the students have been introduced to empirical studies and the opportunity of using them to evaluate different techniques and methods. The objective of the experiment, from an educational perspective, was that the students should be exposed to an empirical study in software verification and validation at the same time as they were introduced to some of the on-going research in the area.

3.2. Inspection Material

The inspection experiment is based on material developed for a verification and validation course in software engineering at Lund University in Sweden. The material consists of four documents in structured text: one requirements document, one design document, one use case document, and one checklist. The use case document and the design document were used in a previous experiment at Lund University. The requirements document and the checklist were developed for this experiment.

The requirements document is written in natural language (English). The document is used as a reference document to know how the system is meant to work. The checklist consists of 18 check items and is based on a checklist presented by Laitenberger et al. [10]. It would have been preferable to use a checklist from an industrial application to check this kind of design, but no such checklist was found. Therefore, we used a modified version of a checklist utilised in experiments with the purpose of comparing CBR and PBR.

The subjects inspected the design document using the requirements document as a reference. To guide the reading they used either a use case document or a checklist. The design consists of software modules of a taxi management system and descriptions of signals in-between these modules. The modules are one taxi module for each vehicle, one central module for the operator and one communication link. The use cases are written in Task Notation [11] and are prioritized using the Analytical Hierarchy Process (AHP) [8] from a user's point of view, i.e. the function of the first use case is the most important to the user while the last use case is least important.

The design document contained 38 faults, of which two were new faults found during the experiment. The 36 others were faults made during development of the design document and later found in inspection or test. These faults were re-inserted prior to the experiment.

The development of the documents and the design of the experiments have involved six persons in total. The persons have taken different roles in the development of the experiment package since it was important to develop and design some parts of the experiment independently in order to minimise the threats to the validity of the experiment. A detailed description of the development of the documents is provided by Thelin et al. [24].

3.3. Fault Classification

The faults are divided into three classes depending on the importance for the user, which is a combination of the probability of the fault to manifest as a failure, and the severity of the fault considered from the user's point of view.

- **Class A faults** – The functions affected by these faults are crucial for the user, i.e. the functions affected are important for the user and are often used. An example of this kind of faults is: the operator cannot *log in* to the system.
- **Class B faults** – The functions affected by these faults are important for the user, i.e. the functions affected are either important and rarely used or not as important but often used. An example of this kind of fault is: the operator cannot *log out* of the system.
- **Class C faults** – The functions affected by these faults are *not* important for the user. An example of this kind of fault is: a signal is confounded with another signal in an MSC diagram.

The design document contains 13 *class A faults*, 14 *class B faults* and *11 class C faults*. No syntax errors like spelling errors or grammatical errors are logged as faults. If these kinds of errors are found, they are not included in the analysis. Three persons made the classification of the faults prior to the experiment.

4. Experimental Planning

4.1. Variables

Three types of variables are defined for the experiment, *independent*, *controlled* and *dependent* variables. The independent variable is the reading technique used and the controlled variable is the experience of the students. The dependent variables are measures collected to evaluate the effect of the methods.

4.2. Hypotheses

The hypothesis of the experiment is that UBR is more efficient and effective in finding faults of the most critical fault classes, i.e. UBR is assumed to find more faults per time unit, and to find a larger share of the critical faults.

The independent variables are analysed to evaluate the hypotheses of the experiment. The main alternative hypotheses are stated below [14]. These are evaluated for all faults, class A faults and class A&B faults. The hypotheses concerns efficiency, effectiveness and fault detecting differences:

- **H_{Eff}** – The reviewers applying use cases are more *efficient* in detecting faults than the reviewers using a checklist, i.e. find more faults per hour.
- **H_{Rate}** – The reviewers applying use cases are more *effective* in detecting faults than the reviewers using a checklist, i.e. find higher rate of total number of faults.
- **H_{Fault}** – The reviewers applying use cases *detect different faults* than the reviewers using a checklist.

4.3. Design

The students were divided into two groups, one group using UBR and one group using CBR. Using the controlled variable to get a block design, the students were divided into three groups and then randomized within each group, resulting in 11 students in the UBR group and 12 students in the CBR group.

The experiment data are analysed with descriptive analysis and statistical tests [21]. The collected data were checked for normal distribution. Since no such distribution could be demonstrated using normal probability plots and residual analysis, nonparametric tests are used. Mann-Whitney [23] is used to investigate hypotheses H_{Eff} and H_{Rate} and a chi square test is used to test H_{Fault}.

4.4. Threats to Validity

The threats to the validity of the experiment are considered under control. As the purpose of the study is to compare two reading techniques, and more studies are needed for generalization purposes, the threats to internal and construct validity are most critical. When trying to generalize the results to a more general domain, the external validity goes more important [26].

Threats to *conclusion validity* are considered under control. Robust statistical techniques are used, measures and treatment implementation are considered reliable. The only risk in the treatment implementation is that the subjects were trained one day and the experiment was conducted the next day. Hence, they might inform each other about the other technique, even though they were strictly forbidden to do. However, nobody would gain from doing so and hence the risk of doing it is low. Random variation in the subject group is blocked, based on the controlled variable.

Concerning the *internal validity*, the risk of rivalry between groups is considered the largest one. However, the student subjects were informed that they would be given additional training in the other reading technique used on the second day. Further, their grade on the course was not affected by the performance in the experiment, only on their attendance.

Threats to the *construct validity* are neither considered very harmful. The development of the textual requirements document was performed after the development of the use cases. Hence, there is a risk that the use cases may have affected the requirements document to make it suitable for the use cases. On the other hand, the inspection object was the design document and the requirements document was just a reference.

Concerning the *external validity*, the use of students as subjects is a threat. However, the students are fourth year master students in software engineering, and a large share of the students have part time jobs in software companies, hence being more representative of software industry than students in general. Further, the size of the inspected document is in the smaller range for real-world problem, even though it describes a real-world problem.

5. Experimental Operation

The experiment was run over two days during spring 2001, see schedule in Table 1.

Table 1: Schedule for the Experiment.

	CBR group	**UBR group**
Day 1 (1.15 p.m. - 2.00 p.m.)	General introduction to the Taxi Management System	
Day 1 (2.15 p.m. - 3.00 p.m.)	Introduction to CBR	Introduction to UBR
Day 2 (9.15 a.m. - 12.00 p.m.)	Inspection Experiment	
Day 2 (1.15 p.m. - 2.00 p.m.)	Introduction to UBR and follow-up discussion	Introduction to CBR and follow-up discussion

6. Analysis

This section presents the data collected during the experiment and pinpoints important issues to discuss in Section 7, where the results are discussed. First, a descriptive analysis is carried out and then the statistical analyses are presented.

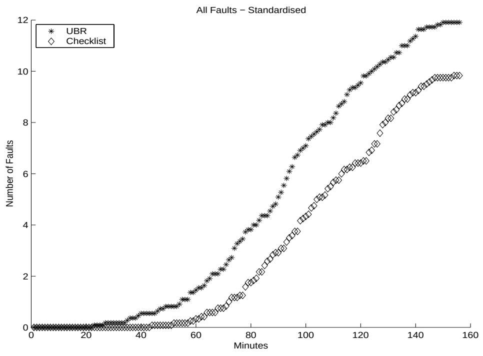

Figure 1. The cumulative number of faults found during the inspection. The data are standardised by the number of reviewers in each group.

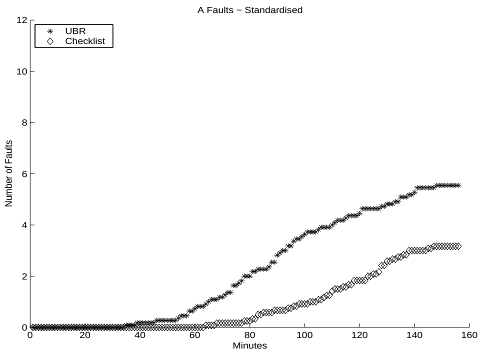

Figure 2. The cumulative number of class A faults found during the inspection. The data are standardised by the number of reviewers in each group.

6.1. Time versus Faults

When the reviewers found a fault, they logged the clock time in the inspection protocol. In Figure 1, the cumulative fault detection is shown. The plot is standardised with respect to the number of reviewers in each groups, i.e. it represents an "average reviewer". The mean preparation time for the UBR group and the CBR group was 53 and 59 minutes respectively.

The reviewers in the UBR group started to find faults earlier than reviewers in the CBR group. The difference is about 20 minutes. Reviewers in the UBR group started to find faults after 54 minutes and the CBR group started to find faults after 74 minutes. (Note that these are a mean values when one fault has been found). This difference could either be due to that it is easier to start with use cases than with a checklist item, or due to that reviewers in the UBR group spend shorter time reading through the documents.

In Figure 2, class A faults are plotted versus detection time and standardised with the number of reviewers. It took a little longer time before the identification of class A faults started and the difference between the UBR and CBR with respect to when the first class A fault found is more than 20 minutes.

In both figures, the slope of the UBR curve increases faster than the curve for the CBR group. This indicates that more severe faults are found more efficiently as well as effectively by reviewers in the UBR group.

In Table 2, the actual figures are presented of how many more faults an "average reviewer" found, i.e. total number of faults found by each group, standardised with the size of the group. A UBR reviewer is more efficient and effective than a CBR reviewer for all classes of faults except for class C faults. A reviewer applying UBR found 75% more class A faults than a CBR reviewer. A similar pattern is shown for class B faults. However, for class C faults, CBR reviewers found most number of faults.

Table 2: A comparison of the number of faults found.

	More faults found	**More Unique Faults**
All Faults	21.1%(UBR)	10.0% (UBR)
Class A Faults	75.1%(UBR)	18.2% (UBR)
Class B Faults	27.7% (UBR)	20.0% (UBR)
Class C Faults	62.5% (CBR)	12.5% (CBR)
Class A&B Faults	50.5% (UBR)	19.0% (UBR)

In Table 2, it is also reported which technique found most unique faults, and the percentage figures show how much more is found compared to the other technique. For all faults, class A and class B faults, the UBR group found more unique faults. They found 18% more unique class A faults and 20% more class B faults. However, CBR found 13% more unique C faults. In total, UBR missed 13% of the faults and CBR missed 21% of the faults**.**

6.2. Effectiveness and Efficiency

The most important characteristic of a reading technique is whether it is efficient and effective enough. Efficiency is defined as the number of faults found per minute and effectiveness is defined as the rate of the total number of faults found in the inspected document. This section provides boxplots of these parameters together with statistical analysis of the performance.

In Figure 3 and Figure 4, the efficiency and the effectiveness are shown. UBR is more efficient as well as more effective than CBR. These figures show the same facts as

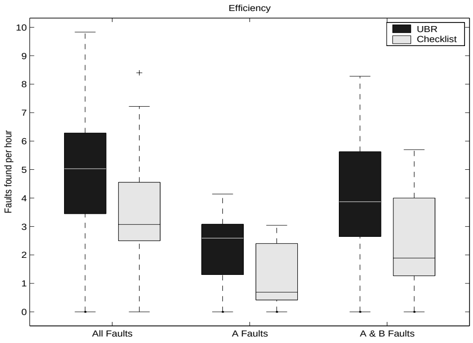

Figure 3. The efficiency for all faults, class A faults and class A&B faults.

discussed earlier in this section. This is true for all faults, class A faults and A&B faults.

In Table 3, p-values for the nonparametric Mann Whitney test are shown. The UBR group is significantly more efficient than the CBR group for class A, class B, class A&B faults. The group is also significantly more effective for class A and class A&B faults. For the rest of the classes, no significant differences can be demonstrated. Hence, according to the statistical analysis combined with the descriptive analysis show that using UBR is significantly more efficient and effective with respect to severe faults.

Table 3: P values for Mann Whitney tests of Efficiency and Effectiveness (α=0.05**).**

	Efficiency (P value)	Effectiveness (P value)
All Faults	0.0423	*0.1029*
Class A Faults	0.0127	0.0364
Class A & B Faults	0.0164	0.0312
Class B Faults	*0.1481*	*0.1754*
Class C Faults	0.2679	0.1481

To test whether the two groups found different faults (H_{Fault}), a Chi-square test is used [23][20]. The test p-value is equal to 0.001, which means that they find different faults in the two groups.

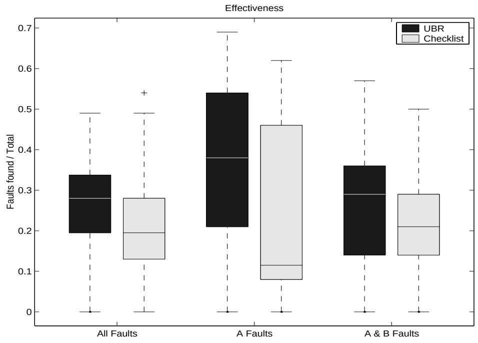

Figure 4. The effectiveness for all faults, class A faults and class A&B faults.

6.3. Team Performance

Although individuals perform inspections, the combined results of an inspection team are the important outcome of an inspection session. Since the reviewers may find the same faults, they may not add as much to the team performance [18]. In order to investigate the reading techniques compared, simulation of the inspection meeting is performed (nominal teams). The purpose of the simulation is to investigate whether a UBR team, a CBR team or a mixed team is the best alternative when performing inspections. The purpose is not to find the ultimate team size, but to analyse the composition of a team. In order to find the best team, trade-off analysis between efficiency and effectivenesss has to be done, which is out of scope of this paper.

To investigate the team performance, all possible combinations were made and the result is shown in Figure 5 and Figure 6. The boxplots show combinations of reviewers only in the UBR group, only in the CBR group and a combination of reviewers in both groups. For example, for the two-inspection-teams, one reviewer from respectively group is used in the mixed teams. Since we have shown that the groups detect different faults, these mixed teams could give better results.

For all team sizes, UBR teams perform better than CBR teams. UBR teams outperform the mixed teams in all cases except for effectiveness in team sizes 5 and 6. However, there are only small differences.

Similar results are obtained when class A faults and class A&B faults are observed in Figure 7 to Figure 8. However, in these cases, the UBR team is better than the mixed team for all team sizes.

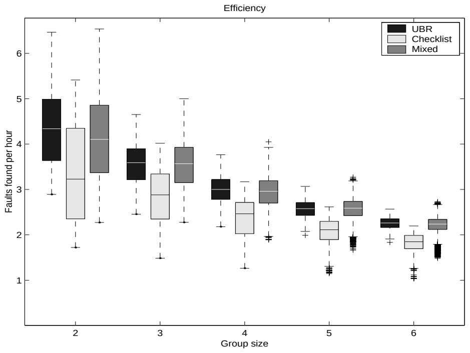

Figure 5. Efficiency for different team sizes. All faults are included.

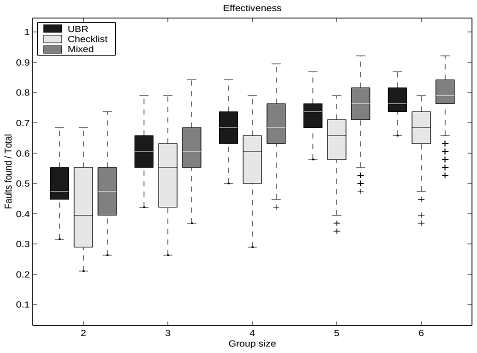

Figure 6. Effectiveness for different team sizes. All faults are included.

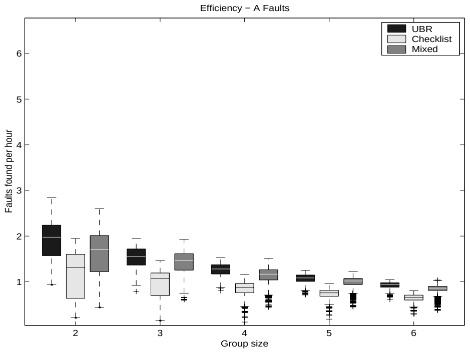

Figure 7. Efficiency for different team sizes. Class A faults are included.

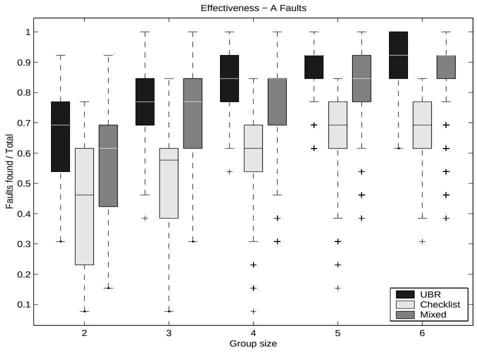

Figure 8. Effectiveness for different team sizes. Class A faults are included.

7. Discussion

The analysis of the experiment results is summarised in Table 4.

Table 4: Summary of the results of the hypotheses.

Group / Fault class	All A+B+C	A	A+B
Efficiency	P=0.042	P=0.013	P=0.016
Effectiveness (Rate)	*P=0.103*	P=0.036	P=0.031
Different (Fault)	P=0.001	–	–

The result can be interpreted as reviewers using UBR are more efficient and effective than reviewers using CBR. They are significantly more *efficient* for all faults and for critical faults. They are more *effective* for critical faults but not for all faults. The assumption when designing the experiment was that they would perform better for critical faults, but not necessary for all faults. The result also shows that reviewers using UBR find different faults than reviewers using CBR.

The team performance analysis shows that it is more efficient to use only UBR reviewers. Pure UBR teams are compared to pure CBR teams and also with mixed teams. Although they find different faults, the mixed teams do not outperform the UBR teams. This can be interpreted as reviewers using UBR are so much better that a combination will not help. However, in some cases a small improvement can be observed in terms of effectiveness.

The UBR reviewers start to find faults earlier than the CBR reviewers do. For all faults, an “average UBR reviewer” starts to find faults 20 minutes before an “average CBR reviewer”. This time increases when critical faults are inves1tigated. This depends on that they spend less preparation time to read through the documents. Even if the reviewers using UBR spend less time in both preparation and inspection, they find significantly more faults. The main explanation for this is probably that the use cases help them to focus on the most important parts in the documents.

The use cases used in UBR are prioritized according to the rank-based reading method. They are prioritized from a user's perspective, since the purpose is to locate the critical faults from a user's point of view. The checklist was not prioritized according to this principle, since a checklist is not function oriented as a use case document. However, the checklist items were ordered in significance order before it was handed out to the students. Furthermore, the CBR group had the opportunity to use the exact same checklist during the *introduction to CBR* (see Table 1) the day before the actual experiment. The UBR group did not see their use cases before the experiment, since the use cases are different for different systems.

Because of the above stated differences between the reading methods, the reviewers in the UBR group find significantly more critical faults and perform in total better than the CBR group. Reviewers using CBR find more class C faults, though not significantly more. This could depend on when inspecting a document with a checklist it is easier to focus on details and more difficult to inspect abstract material, which is necessary in order to find severe faults. During the follow-up session, some students said that it would be beneficial to use both the checklist and the use cases. A hybrid of the UBR method would then beneficial, but we think the checklist need to be adapted for this purpose in order to use it in combination with the use cases.

The results presented by Thelin et al. [24] show that it is possible to guide reviewers more efficiently and more effectively by prioritizing the use cases in the UBR method. In this experiment, the UBR method is baselined against CBR and the study shows positive results. The results are positive from a research perspective but also from an industrial perspective. Especially software organisations using use cases in their development should be interested in the results.

As always when conducting experiments to increase the body of knowledge, the experiment has to the replicated in different contexts. The method should also be investigated in a case study in an industrial setting in order to show whether it still provides positive effects. It would be especially interesting to investigate the method with professionals as subjects.

In order to develop the method further, an experiment investigating time-controlled reading should be conducted. If it is possible to control the reviewers' time consumption and thereby focus only on the critical faults, this could be the starting point of a new important reading technique. A hybrid of UBR and CBR should also be investigated. Applying each use case with some check items or all use cases with a general checklist, the method could be even further improved.

The UBR technique does not produce an artefact during inspection, in contrast to PBR. An experiment to investigate whether reviewers find more faults when actively producing something compared to UBR should be conducted. The experiment would be a comparison of the rank-based method of UBR against the user perspective in PBR. The conclusion of the experiment would be whether it is better to actively produce use cases or passively apply use cases that another person has developed.

8. Summary and Conclusions

The presented experiment compares two reading techniques in order to baseline the rank-based usage-based reading method against the standard industry practice of checklist-based reading. UBR showed promising results in an earlier experiment [24] and even more promising results in this experiment.

The main results from the analysis are that reviewers using UBR find more critical faults and do it more efficiently. The fault severity is defined from a user's point of view. The important results from the experiment are:

- **Efficiency** – Reviewers using usage-based reading are significantly more efficient than reviewers using checklist-based reading. This difference is significant for all faults and for critical faults.
- **Effectiveness** – Reviewers using usage-based reading are significantly more effective than reviewers using checklist-based reading. This difference is significant for critical faults, but not for all faults.
- **Faults** – Reviewers using usage-based reading find different and more unique faults and especially more critical faults than reviewers using checklist-based reading.
- **Teams** – The team analysis also shows that UBR is more effective and efficient than CBR. This is true for all team sizes ranging from two to six.
- **Fault Finding** – A reviewer applying UBR starts to find faults earlier than a reviewer using CBR. The differences for all faults are about 20 minutes and this difference is even larger for critical faults.

Further work is to further develop the method, either to include checklist items or to investigate the time-based ranking method. Although the results are promising, the method needs to be replicated and compared with, for example, the user perspective in PBR.

Acknowledgement

The authors would like to thank the students for participating in the investigation and Thomas Olsson at the Department of Communication Systems at Lund University for developing the taxi management system. We would also like to thank Christer Svensson for work on the requirements specification. Thanks also to Dr. Björn Regnell and Johan Natt och Dag at the Department of Communication Systems for prioritizing the use cases and to Håkan Petersson at the Department of Communication Systems for reviewing an earlier draft of this paper. This work was partly funded by The Swedish National Board for Industrial and Technical Development (NUTEK), under grant for Center for Applied Software Research at Lund University (LUCAS).

References

[1] Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S. and Zelkowitz, M. V., "The Empirical Investigation of Perspective-Based Reading", *Empirical Software Engineering: An International Journal,* 1(2):133-164, 1996.

[2] Bisant, D. B. and Lyle, J. R., "A Two-Person Inspection Method to Improve Programming Productivity", *IEEE Transactions on Software Engineering,* 15(10):1294-1304, 1989.

[3] Ebenau, R. G. and Strauss, S. H., *Software Inspection Process,* McGraw-Hill, New York, 1994.

[4] Eick, S. G., Loader, C. R., Long, M. D., Votta, L. G. and Vander Wiel, S., "Estimating Software Fault Content Before Coding" *Proc. of the 14th International Conference on Software Engineering*, pp. 49-65, 1992.

[5] Fagan, M. E. "Design and Code Inspections to Reduce Errors in Program Development", *IBM System Journal*, 15(3):182-211, 1976.

[6] Gilb, T. and Graham, D. *Software Inspections*, Addison-Wesley, UK, 1993.

[7] Jacobson, I., Christerson, M., Jonsson, P. and Övergaard G. *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, USA, 1992.

[8] Karlsson, J. and Ryan, K., "A Cost-Value Approach for Prioritizing Requirements", *IEEE Software*, 14(5):67-74, 1997.

[9] Knight, J. C. and Myers, A. E., "An Improved Inspection Technique", *Communications of ACM*, 36(11):50-69, 1993.

[10] Laitenberger, O., Atkinson, C., Schlich, M. and El Emam, K., "An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents", *Journal of Systems and Software*, 53(2):183-204 2000.

[11] Lauesen, S., *Software Requirements – Styles and Techniques*, Samfundslitteratur, Denmark, 1999.

[12] Linger, R. C., "Cleanroom Process Model", *IEEE Software,* 11(2):50-58, 1994.

[13] Martin, J. and Tsai, W. T., "*N*-Fold Inspection: A Requirements Analysis Technique", *Communications of ACM,* 33(2):225-232, 1990.

[14] Montgomery, D., *Design and Analysis of Experiments,* John Wiley & Sons, USA, 1997.

[15] Musa, J. D., *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, McGraw-Hill, USA, 1998.

[16] Olofsson, M. and Wennberg, M., "Statistical Usage Inspection", Master's Thesis, Dept. of Communication Systems, Lund University, CODEN: LUTEDX (TETS-5244)/1-81/(1996)&local 9, 1996.

[17] Parnas, D. L. and Weiss, D. M., "Active Design Reviews: Principles and Practices", *Proc. of the 8th International Conference on Software Engineering*, pp. 418-426, 1985.

[18] Petersson, H., Wohlin, C. and Aurum, A., "Team Size and Effectiveness in Software Inspections", submitted to the *Workshop on Inspection in Software Engineering*, 2001.

[19] Porter, A., Votta, L. and Basili, V. R., "Comparing Detection Methods for Software Requirements Inspection: A Replicated Experiment", *IEEE Transactions on Software Engineering,* 21(6):563-575, 1995.

[20] Regnell, B., Runeson, P. and Thelin, T., "Are the Perspectives Really Different? - Further Experimentation on Scenario-Based Reading of Requirements", *Empirical Software Engineering: An International Journal*, 5(4):331-356, 2000.

[21] Robson, C., *Real World Research,* Blackwell, UK, 1993.

[22] Shull, F., Ioana, R. and Basili, V. R., "How Perspective-Based Reading Can Improve Requirements Inspections", *IEEE Computer*, 33(7):73-79, 2000.

[23] Siegel, S. and Castellan, N. J., *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, Singapore, 1988.

[24] Thelin, T., Runeson, P. and Regnell, B., "Usage-Based Reading – An Experiment to Guide Reviewers with Use Cases", to appear in *Information and Software Technology,* 2001.

[25] Weller, E. F., "Lessons from Three Years of Inspection Data", *IEEE Software,* 10(5):38-45, 1993.

[26] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. and Wesslén, A., *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publisher, USA, 2000.