Using Inspections to Teach Requirements Validation

Lulu He, Dr. Jeffrey C. Carver, and Dr. Rayford B. Vaughn
Mississippi State University

Requirements validation is often not adequately covered by a traditional software engineering curriculum in universities. This article describes an experiment conducted in a graduate-level requirements engineering course to provide students a real world experience in requirements validation. The experiment made use of the N-fold inspection method, in which multiple teams of students inspect the same requirements document then meet together to discuss their findings. This procedure allows the students to not only practice their reviewing skills, but also to strengthen their communication and collaboration skills. At the conclusion of the exercise, the students were given the opportunity to provide qualitative and quantitative feedback. The results of this study suggest that the techniques employed by this class and the resulting defect detection could be useful in general during the requirements validation process.

It can be argued that requirements engineering (RE) is one of the most important stages in a traditional software development life cycle because it helps to correctly determine the desired purpose of a system. The goal of an RE process is to identify and document stakeholders and their needs in a form amenable to analysis, communication, and implementation [1]. Correct, complete, and unambiguous requirements not only ensure that developers build the right system, but also reduce the effort and cost that would otherwise be needed to fix requirements problems later. Because of the importance in obtaining correct requirements, RE courses are fundamental elements of any software engineering curriculum. An RE course should teach students techniques for accurately eliciting, analyzing, and validating requirements. By acquiring these skills, future software engineers are better equipped to produce quality requirements, eliminate faults, and reduce development time. Educating students in RE, however, is not an easy task because it is a human-centered process that requires skills from a variety of disciplines (e.g., computer science, system engineering, cognitive psychology, anthropology, and sociology) [1]. Moreover, RE educators must bridge the gap between academia and industry by exposing students to the real world as much as possible. One important real-world aspect of RE that has remained largely unstudied in the software engineering education community is requirements validation.

This article describes an exercise to aid in the teaching of requirements validation in a graduate-level RE course, along with an evaluation of its usefulness. In the exercise, the students validated a real software requirements specification document provided by an industrial partner using a meeting-based N-fold inspection method (described later) [2].

Background

Previous research has identified challenges faced during RE education. Because one goal of software engineering education is to help students develop the knowledge and skills necessary to be successful in industry, a challenge for RE educators is that the inherent complexity of industrial RE (i.e. the broad scope, multiple concerns, and deficient specifications) is difficult to replicate in a classroom environment [3, 4]. To be successful, a requirements engineer must possess both the technical skills needed to interact with the system and the social skills needed to interact with its human stakeholders [1]. These *soft skills* (e.g., communication and teamwork) are also difficult to teach in the classroom [5].

Most published work on RE education has focused on a small set of RE topics: elicitation, analysis, and the overall process. Validation is a complex task that is often not adequately covered in a traditional university education [6]. A requirements document can have many different types of deficiencies that may have disastrous effects on subsequent development stages and yield undesirable consequences [4, 7]. The requirements validation process checks for different types of problems (e.g., omissions, inconsistencies, and ambiguities) to help ensure that proper quality standards are fulfilled. Due to time limits and other considerations, validation is usually conducted informally either on an ad-hoc basis or simply as a peer review [8]. As a result, little has been reported concerning the educational challenges of validation topics like inspections. These challenges highlight the importance of improving education and critical, technical, and social skills that a requirements engineer must possess. However, the topics covered in many software engineering courses do not meet these needs, posing a major challenge for developing RE skills [6, 9]. Furthermore, experience reports about RE education is rare in software engineering and RE literature. To address the lack of focus on requirements validation, we developed a requirements validation exercise which we found to be successful and instructive. This exercise is a follow-on experiment from that reported at the Fourteenth Annual Systems and Software Technology Conference held in Salt Lake City in April 2002 titled "Third Party Walkthrough Inspections: A Joint Navy/University Empirical Software Engineering Project." The most recent results of the experiment were also presented by invitation at the Canadian Air Force Software Engineering Symposium held at the Royal Military College of Canada in Kingston, Ontario, May 2007 [10, 11].

Description of the Requirements Validation Exercise

The exercise was conducted during a graduate-level RE course at Mississippi State University. The class included graduate students that were currently working for the Department of Defense (DoD), had previous government software development experience, and many that had no practical experience. The main goal of this course was to provide students with a comprehensive overview of requirements elicitation, analysis, specification validation, and management. These activities were introduced in the context of systems engineering and various software development life-cycle models. For each activity, the students were exposed to specific methods, tools, and notations. The semi-weekly, 75-minute class sessions contained a mixture of lecture material, class discussions centered on outside readings, and student presentations. Near the end of the semester, the students participated in a two-week requirements validation exercise – the primary subject of

this article. While the exercise reported here is based on a small number of students, we believe the results achieved suggest that such techniques could be considered in larger requirements validation exercises. The authors are amenable to cooperating with others to expand this research through additional empirical investigation – particularly in DoD software engineering endeavors.

Overview and Goals of the Requirements Validation Exercise

The overall goal of the exercise was to provide students with hands-on practice in requirements validation. The specific goals of the requirements validation exercise were the following:

1. To help students understand the course materials better by experiencing the format and presentation of a real requirements document, experiencing the impact of domain-specific language in understanding and reviewing a requirements document, and exposing the students to flaws commonly found in a requirements document.
2. To help students obtain an appreciation for the complexity and necessity of RE, especially requirements validation.
3. To give students hands-on experience validating a requirements document with specific inspection techniques.
4. To provide an opportunity for students to practice *soft skills* such as communication and teamwork.

Two important goals of the course were addressed through this exercise. First, give the students an idea of the size, complexity, language, and flaws that can occur in software artifacts, they validated a real requirements document, which was actually used by a contractor to develop and implement a system. The requirements document described an upgrade to a case-tracking system for the U.S. National Labor Relations Board. The 43-page document was written in natural language (English) and contained the standard content that would be expected in a government requirements document.

Second, to expose students to a real-world requirements validation method, the *N-fold inspection method* was used. In this method, the same artifact is inspected by multiple teams in parallel with the goal of improving the overall review effectiveness [12]. From an educational point of view, the N-fold inspection method provides students with an opportunity to discuss the defects found with other students, thereby understanding how others had viewed the artifact. Furthermore, N-fold inspection meetings expose students to the importance of communication and teamwork – important soft skills.

Requirements Validation Techniques

Studies have shown that inspections are an effective requirements validation technique because they greatly improve system quality by detecting many defects early in the software life cycle [13]. Within the N-fold method, individual reviewers can use different approaches to review the document. Our students used either a checklist approach or the Perspective-Based Reading (PBR) approach (each described in more detail).

In practice, most industrial inspections use an ad-hoc or checklist-based approach for defect detection [14]. A checklist provides the inspector with concrete guidance that is not provided by an ad-hoc approach. A checklist is a list of items that focus an inspector's attention on specific topics, such as common defects or organizational rules, while reviewing a software document [15]. For this requirements validation exercise, an informal checklist was developed that focused on important quality concepts relevant to a requirements document. Checklist examples are readily available on the Web, for example, <software.gsfc.nasa.gov/AssetsApproved/PA2.2.1.5.doc>, <www.processimpact.com/process_assets/requirements_review_checklist.doc>, or <www.swqual.com/training/Require.pdf>. For the class exercise, we used a checklist based on the Volere Requirements Specification Template which can be found at <www.systemsguild.com/GuildSite/Robs/Template.html>.

> *"Studies have shown that inspections are an effective requirements validation technique ... by detecting many defects early in the software life cycle."*

PBR is a systematic inspection technique that supports defect detection in software requirements through a role-playing exercise [13]. In PBR, each reviewer verifies the correctness of the requirements from the perspective of a specific stakeholder. The most common perspectives are the user, the designer, and the tester. PBR techniques provide reviewers with a set of steps to follow to build a high-level system abstraction and questions to help identify problems. For example, a reviewer assuming the user perspective may create use cases, while a reviewer assuming the tester perspective would create test plans. Studies have shown that PBR is a more effective, systematic, focused, goal-oriented, customizable, and transferable process than a standard checklist or ad-hoc approach [16]. By reviewing the requirements document from the perspective of different stakeholders (i.e., playing the role of that stakeholder), the reviewers are expected not only to detect more defects, but also to better comprehend the complexity of RE.

Details of the Requirements Validation Exercise and Summary of Research Results

Students enrolled in the course were divided into four, three-person teams with the expertise level balanced across teams. The members of two teams used a checklist to inspect the requirements document, while members of the other two teams used the PBR technique. For the PBR teams, each student was instructed to use one of the three perspectives (user, designer, or tester) to guide their inspection. These two inspection techniques were chosen because research suggested that while PBR is often a more effective technique, checklists are more widely used in government and industry [14]. A second motivator was that no previous work had compared the effectiveness of a checklist-based approach to the effectiveness PBR in the context of the N-fold inspection method. More importantly, instead of using a generic checklist, the checklist in this study was specifically developed for the type of requirements document to be inspected (e.g., from a government organization).

Before the exercise began, the students received one class meeting (75 minutes) of training in their assigned technique (checklist or PBR). The training for the checklist teams was done through a discussion with the course professor about attributes of requirements quality. During this discussion, the checklist students – heavily guided by the professor – developed a checklist to guide their inspection. The training for PBR was done by an expert in PBR and included a discussion of the theory behind the techniques and a case study that illustrated an example of its use. After the training, the

PBR reviewers were given the detailed protocol for their assigned perspective. Then each student performed an individual inspection of the requirements document using their assigned technique. During this inspection, each student individually reviewed the document and recorded as many defects as he or she could find. The students were given two days to perform this task outside of class. Once all three members of a team completed the individual inspection step, they met together in the 1st Team Meeting. During this meeting, the students discussed the defects they found and agreed on a final team defect list. After all four teams had conducted the 1st Team Meeting, the two checklist teams (six students) met together and the two PBR teams (six students) met together for the 2nd Team Meeting. In these six-person meetings, the reviewers examined the two defect lists produced during the 1st Team Meeting and agreed on a final list of defects.

The data analysis indicated that PBR was more effective, both for individual inspectors and for the teams. Conversely, the data suggested that the checklist teams had more effective team meetings during the N-fold inspection process than the PBR teams. Here, the effectiveness of team meetings is defined by two measures: *meeting gains*, i.e. the number of defects identified during the meeting discussions that no individual inspector had found prior to the meeting, and *meeting losses*, i.e. defects found by an individual inspector that the inspection team determines are false positives and, hence, not recorded on the final team defect list. In the case of the PBR teams, the team meeting served little purpose because there were few meeting gains or meeting losses. The end result would have been similar had the individual list simply been combined without spending time in a formal team meeting. Conversely, for the checklist teams, the meeting served a vital role in the process. During the team meetings, not only were there meeting gains, but also a large percentage of false positives were identified and removed as meeting losses. Identification of false positives saves time during the rework phase. One likely cause of this result is the different perspectives from which the PBR reviewers approached the Software Requirements Specification (SRS). Each PBR reviewer focused on their own perspective and was less concerned with the perspectives of others. For the checklist team, the reviewers inspected the SRS using the same checklist and there was more interaction among team members during the meetings. The most important, and novel, conclusion drawn from these results is that the effectiveness and necessity of a team meeting depends greatly on the technique used during the individual preparation phase of the inspection. Additional data on this experiment can be obtained by contacting the second or third author of this article.

Evaluation of the Educational Value of the Exercise

To evaluate the effectiveness of this exercise relative to the four educational goals listed earlier, quantitative and qualitative data was collected using a post-study survey. Goals 1-3 were specifically evaluated by the survey questions in Table 1 (see page 14). Goal 4 was addressed by using team meetings, but it was not specifically evaluated on the post-study questionnaire. The survey focused on the students' opinions of the exercise and gave them an opportunity to provide feedback on how to improve the exercise. The first two questions were answered using a predetermined scale (explained with each question). The other questions were answered using free text.

Results

This section discusses the results from the post-exercise questionnaire along with a brief explanation. The questionnaire collected both qualitative and quantitative data from the students.

Quantitative Data

Q1. Was the exercise a positive or negative experience?

The students answered this question on a scale of 1 (negative) to 5 (positive). Figure 1 shows that 91 percent of the students found the exercise to be a positive experience with no students having a negative experience. In Figure 1, the ratings given by students using PBR and checklist are shown in different shades to evaluate any impact the technique had. While the opinion of the PBR reviewers is slightly more positive, there is little difference due to the technique used. Therefore, regardless of which technique was used, the students found the N-fold inspection exercise to be a positive experience.

Q2. Did the exercise help you better understand the course materials?

The students responded to this question on a scale of 1 (not at all) to 5 (very much). Figure 2 shows that all of the students found the exercise was *very helpful* or a *lot* helpful.

Qualitative Data

The qualitative feedback, in which the students were able to express their own opinions, provides additional insight into the usefulness and effectiveness of the exercise. In this analysis, specific answers given for each question are listed along with the number of students who gave that answer, shown in parentheses where *3 checklist/2 PBR* means that three students who used the checklist and two students who used PBR gave the answer. In some cases, the response was given by only PBR students or only checklist students, e.g., *3 PBR* in bullet three of Q3. This does not imply that the answer is not applicable to the other technique – it only means that no students using the other technique provided an answer. In some cases, students provided more than one answer to each question, so the total number of responses may be greater than 12. Q3 and Q4 from the post-study survey were geared towards addressing Educational Goals 2 and 3. Q5

Figure 1: *Positive or Negative Experience Results*

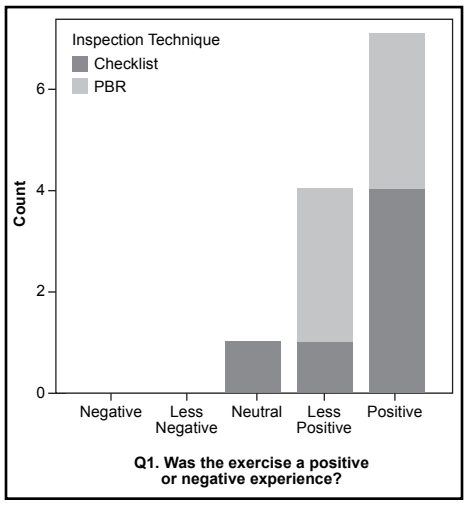
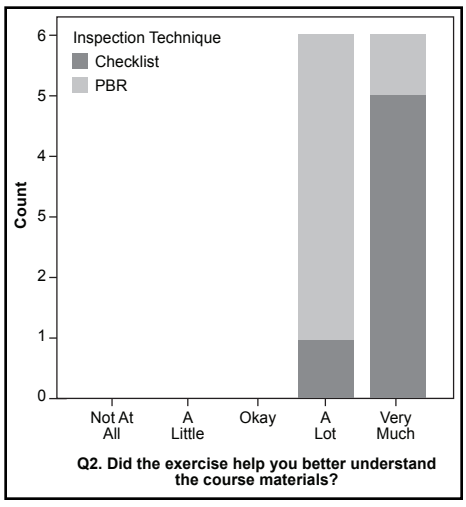


Figure 2: *Understanding the Course Materials Results*



Q1. Was the exercise a positive or negative experience?
Q2. Did the exercise help you better understand the course materials?
Q3. Did you see any benefits from the exercise? If so what were they?
Q4. Did you learn anything by performing the exercise? If so, what?
Q5. Did you see any drawbacks to the exercise? If so what were they?
Q6. How could the exercise been improved?

Table 1: *Survey Questions*

and Q6 provide feedback on how to improve the exercise.

Q3. Did you see any benefits from the exercise? If so what were they?

All 12 students indicated that they found some benefit from participation in this exercise:

1. It provided hands-on/practical experience (3 checklist/2 PBR).
2. It helped them better understand a real requirements document (3 checklist /3 PBR).
3. It helped them better understand defect detection in a requirements document (3 PBR).

Because the students obtained hands-on experience and indicated that they understood a real requirements document, this exercise helped address Educational Goals 2 and 3.

Q4. Did you learn anything by performing the exercise? If so, what?

From this exercise, the students learned the following:

1. The usefulness of inspections (2 checklist/1 PBR).
2. The difficulties involved in creating and using a real requirements document (3 checklist/1 PBR).
3. The benefits of using a method to focus a requirements inspection on certain aspects of the requirements document (2 checklist/5 PBR).

Similar to Q3, these responses indicate that the students learned the benefits of inspections, and the difficulties in creating real requirements documents, helping address Educational Goals 2 and 3.

Q5. Did you see any drawbacks to the exercise? If so what were they?

Only five (4 checklist/1 PBR) of the 12 students reported any drawbacks of the exercise:

1. Not enough training (3 checklist/1 PBR).
2. Not enough time (2 checklist).

On a positive note, these drawbacks all relate to the logistics of the exercise and not to its intrinsic value. None of these drawbacks are concerned with the inspection procedure that was used during the exercise.

Q6. How could the exercise have been improved?

1. Expand the exercise (1 checklist/1 PBR).
2. Provide more domain knowledge (2 checklist).
3. Allow more time for various activities (2 checklist/2 PBR).
4. Provide more training (1 checklist/4 PBR).

These suggestions generally relate to the drawbacks cited in Q5. It was interesting that two students asked for a more extensive exercise that allowed them to obtain more practice and experience using the inspection techniques. This request suggests that the students believed that even more benefit would be obtained by expanding the scope of the exercise.

Summary and Conclusion

This article describes the use of a requirements validation exercise in a graduate-level RE course. In the exercise, the students validated a software requirements document using a meeting-based N-fold inspection. The students provided their opinions and feedback about the usefulness of the exercise in a post-exercise survey. These results showed that the exercise achieved its goals.

Overall, students were highly satisfied with the exercise content and found it to be helpful. The exercise helped students understand what a software requirements document looks like and gain insight into the difficulty and complexity involved in its correct development. They not only realized the importance of requirements validation but also gained hands-on experience using inspection techniques. The exercise provided the students with essential knowledge about requirements quality, enabling them to better understand other course material such as the following: elicitation, analysis, and specification. Moreover, though not empirically evaluated, the team meetings in this exercise gave the students an opportunity to practice their communication and teamwork skills, which are essential for their future careers.

Based on the feedback provided by the students, the following modifications are being considered for future similar class exercises:

1. Provide more training on the inspection techniques by using case studies.
2. Give specific instructions on the structure and organization of the team meeting.
3. Extend the length of the exercise to allow additional time for training and the individual inspection.
4. To make the exercise more realistic, stakeholders of the software requirements document will be invited to class, and help students to obtain more domain knowledge.
5. When the N-fold inspection is finished, the defect lists will be given to the owner of the requirements document to understand the disposition of those defects.

This study was performed in a graduate-level university course; therefore, the next step is to perform additional validation in an industrial setting. Furthermore, the positive experience with the N-fold inspection process during RE suggests that it may have applicability in other phases of the software life cycle. In the future, we will seek out opportunities to replicate these experiences in other aspects of the software engineering curriculum.◆

References

1. Nuseibeh, B., and S. Easterbrook. "Requirements Engineering: A Road Map." Proc. of International Conference on Software Engineering, Limerick, Ireland, June 2000: Association of Computing Machinery (ACM) Press, 2000: 37-46.
2. He, L., and J.C. Carver. "PBR Vs. Checklist: A Replication in the N-Fold Inspection Context." Proc. of 5th ACM/Institute of Electrical and Electronics Engineers (IEEE) International Symposium on Empirical Software Engineering (ISESE 2006), Rio de Janeiro, Brazil, Sept. 21-22, 2006.
3. Armarego, J., and S. Clarke. "Preparing Students for the Future: Learning Creative Software Development – Setting the Stage." Proc. of the Annual International Higher Education Research and Development Society of Austral Asia Conference Christchurch, New Zealand. July 6-9, 2003.
4. Van Lamsweerde, A. "Requirements Engineering in the Year 00: A Research Perspective." Proc. of 22nd

International Conference on Software Engineering. Limerick, Ireland, 2000: IEEE Computer Society Press, 2000.
5. Conn, R. "Developing Software Engineers at the C-130j Software Factory." IEEE Software 19.5 (2002): 25-29.
6. Bubenko, J.A. "Challenges in Requirements Engineering." Proc. of 2nd IEEE International Symposium on Require-ments Engineering, Los Alamitos, CA: IEEE Computer Society, 1995: 160-162.
7. Meyer, B. On Formalism in Specifications. IEEE Software 2.1 (1985): 6-26.
8. Rosca, D. "An Active/Collaborative Approach in Teaching Requirements Engineering." Proc. of 30th Annual Frontiers in Education Conference. Kansas City, MO., Oct., 2000: 9-12.
9. Lethbridge, T.C. "What Knowledge Is Important to a Software Professional?" Computer 33.5 (2000): 44-50.
10. Vaughn, R.B., and J. Lever. "Third Party Walkthrough Inspections: A Joint Navy/University Empirical Software Engineering Project." Proc. of Fourteenth Annual Systems and Software Technology Conference, Salt Lake City, UT, Apr. 29-May 2, 2002.
11. Vaughn, R.B., and J.C. Carver. "Experiences in N-Fold Structured Walkthroughs of Requirements Documents." Proc. of Canadian Air Force Software Engineering Symposium,. Royal Military College of Canada, Kingston, Ontario. 24-25 May, 2007.
12. Martin, J., and W. Tsai. "N-Fold Inspection: A Requirements Analysis Technique." Communications of the ACM 33.2 (1990): 223-232.
13. Basili, V.R., et al. "The Empirical Investigation of Perspective-Based Reading." Empirical Software Engineering: An International Journal 1.2 (1996): 133-164.
14. Laitenberger, O., K.E. Emam, and T.G. Harbich. "An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective-Based Reading of Code Documents." IEEE Transactions on Software Engineering 27.5 (2001): 387-421.
15. Fagan, M. "Design and Code Inspections to Reduce Errors in Program Development." IBM System Journal 15.3 (1976): 182-211.
16. Shull, F., I. Rus, and V.R. Basili. "How Perspective-Based Reading Can Improve Requirements Inspection." IEEE Software 33.7 (2000):73-79.

About the Authors

Lulu He is a doctorate. student in the Computer Science and Engineering Department at Mississippi State University. She received her bachelor's and master's degrees in computer science from Wuhan University, China in 2001 and 2004, respectively. She also received a master's degree in computer science from Mississippi State University in August, 2007. Her research interests include Software Quality, Software Inspections, Software Architecture, and Software Engineering for Scientific and Engineering Computing.

Computer Science
and Engineering
PO Box 9637
Mississippi State University
Mississippi State, MS 39762
Phone: (662) 325-8798
E-mail: lh221@msstate.edu

Jeffrey Carver, Ph.D., is an Assistant Professor in the Computer Science and Engineering Department at Mississippi State University. He received his doctorate degree from the University of Maryland in 2003. His research interests include software process improvement, software quality, software inspections, and software engineering for scientific and engineering computing. He has more than 30 refereed publications in these areas. His research has been funded by the U.S. Army Corps of Engineers, the U.S. Air Force, and the National Science Foundation.

Computer Science
and Engineering
PO Box 9637
Mississippi State University
Mississippi State, MS 39762
Phone: (662) 325-0004
Fax: (662) 325-8997
E-mail: carver@cse.msstate.edu

Rayford B. Vaughn, Ph.D., received his doctorate from Kansas State University in 1988. He is a William L. Giles Distinguished Professor and the Billie J. Ball Professor of Computer Science and Engineering at Mississippi State University and teaches and conducts research in the areas of Software Engineering and Information Security. Prior to joining the university, he completed a 26-year career in the U.S. Army retiring as a Colonel and three years as a Vice President of Defense Information Systems Agency Integration Services, and EDS Government Systems. Vaughn has more than 100 publications to his credit and is an active contributor to software engineering and information security conferences and journals. In 2004, he was named a Mississippi State University Eminent Scholar. Vaughn is the current Director of the Mississippi State University Center for Critical Infrastructure Protection and the Center for Computer Security Research.

Computer Science
and Engineering
PO Box 9637
Mississippi State University
Mississippi State, MS 39762
Phone: (662) 325-7450
Fax: (662) 325-8997
E-mail: vaughn@cse.msstate.edu