Experimenting with Techniques for Defect Detection in Requirements

Victor R. Basili, Forrest Shull, and Marvin Zelkowitz
Institute for Advanced Computer Studies
Computer Science Department
University of Maryland
College Park, MD, USA
{basili, fshull, mvz}@cs.umd.edu

Filippo Lanubile
Dipartimento di Informatica
Universita' di Bari
Via Orabona, 4
70126 Bari, Italia
lanubile@di.uniba.it

1. Purpose of the Experiment

In previous experiments [Basili96], we showed that the Perspective-Based Reading (PBR) family of defect detection techniques was effective at detecting faults in requirements documents in some contexts. In order to continue adding to our knowledge in this area, we have proposed two modifications to the PBR technique that we will assess in the proposed experiment.

The first purpose of the experiment is to investigate whether defect detection in requirements documents can be improved by focusing on the *errors* (i.e. underlying human misconceptions) in a document rather than the individual *faults* that they cause. The rationale for this is that, in our experience, faults have been very difficult to quantify, classify, and define. Additionally, by treating faults as isolated entities there is little possibility for using discovered faults to help find new ones. We are investigating whether the process of abstraction from faults to errors may give reviewers useful information about the requirements that would help them in discovering additional faults and fixing the document. We will evaluate the effectiveness of error abstraction in both *ad hoc* and PBR reviews, in order to examine potential interactions between the abstraction process and the fault detection technique used.

The secondary purpose of the experiment is to investigate the effect of increasing the specificity of the PBR techniques. In the original experiment with PBR, we asked reviewers to rely on their experience with other processes in the software lifecycle (creating high-level designs, test plans, and user manuals) and to use this experience to answer specific questions. We propose changing the PBR techniques to provide detailed guidelines for each process used (e.g. Object-Oriented design, equivalence partitioning, and constructing use cases, respectively) with the specific questions distributed at key points in the guidelines. For novice reviewers, we hope to see that the additional level of specificity may help make up for the lack of experience in these software development processes. For expert reviewers, we hope to see that the additional specificity helps to focus them on the PBR technique instead of relying on their accustomed approach. To us as experimenters, the additional level of detail in the process means that we can now assess the level of process conformance to the PBR technique.

2. Design and Research Hypotheses

In running this experiment in a classroom environment, we are subject to a number of constraints. Most importantly:

- Because all members of the class must be taught something, we cannot have a control group (because we cannot have a segment of the class with no treatment or training).
- Because subjects are not isolated, and would presumably discuss their experiences at various parts of the experiment amongst themselves, we cannot "cross" the experimental artifacts (that is, use a particular artifact for one group before treatment and for another group after treatment).

We therefore use an experimental design which takes the above constraints into account but which can easily be adapted for environments in which the constraints do not hold.

We propose the following design, in which the entire group of subjects moves together through six phases, three of which examine the effect of the error abstraction process followed by team meetings on *ad hoc* reviews, and a further three which repeat the same examination for PBR reviews:

TIME

	Baseline	**Individual Treatment**	**Team Treatment**
Requirements Document 1	*Ad hoc* reviews		
		Error abstraction	
			Discussion of fault and error lists
Requirements Document 2	PBR reviews		
		Error abstraction	
			Discussion of fault and error lists

Thus for each review technique (*ad hoc* and PBR) we can collect and analyze the following process outputs:

1. The initial list of faults discovered
2. The list of abstracted errors, and a revised list of faults that may have been improved based on knowledge of the abstracted errors
3. A final list of faults and errors that result after the teams have discussed and merged the lists of the individual members

We also expect to collect qualitative data about the operation of the process through questionnaires and interviews.

With this information we will evaluate the following hypotheses about the error abstraction process:

- Error abstraction (with and without PBR) results in a significantly different average defect detection rate than if error abstraction is not used.
- Error abstraction (with and without PBR) results in a significantly different average number of false positives reported than if error abstraction is not used.
- Team discussion with error abstraction (with and without PBR) results in a significantly different average defect detection rate than if team discussions are not used.
- Team discussion with error abstraction (with and without PBR) results in a significantly different average number of false positives reported than if team discussions are not used.
- Reviewer experience (in general or in their PBR role) impacts effectiveness at defect detection.
- Unique sets of errors are identifiable for a particular set of requirements.
- The mapping between faults and errors can be characterized.

And these about the increased specificity of the PBR techniques:

- The use of PBR is more effective (i.e., results in a higher average defect detection rate, results in a lower number of false positives reported, and does not require more time) than an *ad hoc* process.
- The use of PBR with error abstraction is more effective (i.e., results in a higher average defect detection rate, results in a lower number of false positives reported, and does not require more time) than the use of error abstraction alone.

- Team meetings with PBR are more effective (i.e., results in a higher average defect detection rate, results in a lower number of false positives reported, and does not require more time) than team meetings in which it has not been used.

3. Expectations Of Collaboration

We are seeking other sites who would be interested in running this experiment, so that through replication some of our particular constraints may be overcome. Collaborating sites would receive versions of all of our techniques which would be improved based on any feedback we get from this initial run of the experiment. Sites who provide developers for use as subjects in the experiment will receive training for those developers in:

- The error abstraction process
- The PBR method for detecting defects in requirements
- The following software processes (one per subject): high-level OO design, equivalence partitioning, use case creation

We will have the detailed techniques available on the WWW at http://www.cs.umd.edu/projects/SoftEng/ESEG/manual/error_abstraction/manual.html as soon as the experiment is over in class. After that, we plan to add sections on the experimental motivation, design, and analysis of the experiment for use by other researchers.

4. References

[Basili96] V. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, M. Zelkowitz, The Empirical Investigation of Perspective-Based Reading, Empirical Software Engineering: An International Journal, vol. 1, no. 2, pp. 133-164, 1996.