

Evolving and Packaging Reading Technologies

V.R. Basili

*Department of Computer Science
and Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742 U.S.A.
(301)405-2668, (301)405-6707 FAX, basili@cs.umd.edu*

Abstract

Reading is a fundamental technology for achieving quality software. This paper provides a motivation for reading as a quality improvement technology, based upon experiences in the Software Engineering Laboratory at NASA Goddard Space Flight Center and shows the evolution of our study of reading via a series of experiments. The experiments range from the early reading vs. testing experiments to various Cleanroom experiments that employed reading to the development of new reading technologies currently under study.

Keywords

Reading scenarios, cleanroom, experiments, inspections, quality improvement paradigm

1. INTRODUCTION

Reading is a fundamental technology for achieving quality software. It is the only analysis technology we can use throughout the entire life cycle of the software development and maintenance processes. And yet, very little attention has been paid to the technologies that underlie the reading of software documents. For example where is “software reading” taught? What technologies have been developed for “software reading”? In fact, what is “software reading”?

During most of our lives, we learned to read before we learned to write. Reading formed a model for writing. This was true from our first learning of a language (reading precedes writing and provides simple models for writing) to our study of the great literature (reading provides us with models of how to write well). Yet, in the software domain, we never learned to read, e.g., we learn to write programs in a programming language, but never learn how to read them. We have not developed reading-based models for writing.

For example, we are not conscious of our audience when we write a requirements document. How will they read it? What is the difference between reading a requirements document and reading a code document? We all know that one reads a novel differently than one reads a text book. We know that we review a technical paper differently than we review a newspaper article. But how do we read a requirements document, how do we read a code document, how do we read a test plan?

But first let us define some terms so that we understand what we mean by “reading”. We differentiate a technique from a method, from a life cycle model. A technique is the most primitive, it is an algorithm, a series of steps producing the desired effect. It requires skill. A method is a management procedure for applying techniques, organized by a set of rules stating how and when to apply and when to stop applying the technique (entry and exit criteria), when the technique is appropriate, and how to evaluate it. We will define a technology as a collection of techniques and methods. A life cycle model is a set of methods that covers the entire life cycle of a software product.

For example, reading by step-wise abstraction [Linger, Mills, and Witt, 1979] is a technique for assessing code. Reading by stepwise abstraction requires the development of personal skills; one gets better with practice. A code inspection is a method, that is defined around a reading technique, which has a well defined set of entry and exit criteria and a set of management supports specifying how and when to use the technique. Reading by stepwise abstraction and code inspections together form a technology. Inspections are embedded in a life cycle model, such as the Cleanroom development approach, which is highly dependent on reading techniques and methods. That is, reading technology is fundamental to a Cleanroom development.

In what follows, we will discuss the evolution and packaging of reading as a technology in the Software Engineering Laboratory (SEL) [Basili, Caldiera, McGarry, Pajerski, Page, Waligora, 1992] via a series of experiments from some early reading vs. testing technique experiments, to various Cleanroom experiments, to the development of new reading techniques currently under study.

In the SEL, we have been working with a set of experimental learning approaches: the Quality Improvement Paradigm, the Goal Question Metric Paradigm, the Experience Factory Organization, and various experimental frameworks to evolve our knowledge and the effectiveness of various life cycle models, methods, techniques, and tools [Basili - 1985, Basili and Weiss - 1984, Basili and Rombach - 1988, Basili - 1989]. We have run a series of experiments at the University of Maryland and at NASA to learn about, evaluate, and evolve reading as a technology.

2. READING STUDIES

Figure 1 provides a characterization of various types of experiments we have run in the SEL. They define different scopes of evaluation representing different levels of confidence in the results. They are characterized by the number of teams replicating each project and the number of different projects analyzed yielding four different experimental treatments: blocked subject-project, replicated project, multi-project variation, and single project case study.

The approaches vary in cost, level of confidence in the results, insights gained, and the balance between quantitative and qualitative research methods. Clearly, an analysis of several replicated projects costs more money but provides a better basis for quantitative analysis and can generate stronger statistical confidence in the conclusions. Unfortunately, since a blocked subject-project experiment is so expensive, the projects studied tend to be small. To increase the size of the projects, keep the costs reasonable, and allow us to better simulate the effects of the treatment variables in a realistic environment, we can study very large single project case studies and even multi-project studies if the right environment can be found. These larger projects tend to involve more qualitative analysis along with some more primitive quantitative analysis.

Because of the desire for statistical confidence in the results, the problems with scale up, and the need to test in a realistic environment, one approach to experimentation is to choose one of the multiple team treatments controlled experiments to demonstrate feasibility (statistical significance) in the small, and then to try a case study or multiproject variation to analyze whether the results scale up in a realistic environment - a major problem in studying the effects

of techniques, methods and life cycle models.

Scopes of Evaluation			
		#Projects	
		One	More than one
# of Teams	One	Single Project (Case Study)	Multi-Project Variation
per Project	More than one	Replicated Project	Blocked Subject-Project

Figure 1: Classes of Studies

2.1 Reading by stepwise abstraction

In order to improve the quality of our software products at NASA, we have studied various approaches. One area of interest was to understand the relationship between reading and testing in our environment. Early experiments showed very little difference between reading and testing [Hetzel - 1972, Myers - 1978]. But reading was simply reading, without a technological base. Thus we attempted to study the differences between various specific technology based approaches. Our goal was to analyze code reading, functional testing and structural testing to evaluate and compare them with respect to their effect on fault detection effectiveness, fault detection cost and classes of faults detected from the viewpoint of quality assurance [Basili, Selby - 1987]. The study was conducted in the SEL, using three different programs: a text formatter, a plotter, and a small database. The programs were seeded with software faults, 9, 6, and 12 faults respectively, and ranged in size from 145 to 365 LOC. The experimental design was a blocked subject-project, using a fractional factorial design. There were 32 subjects.

Specific techniques were used for each of the three approaches studied. Code reading was done by stepwise abstraction, i.e., reading a sequence of statements and abstracting the function they compute and repeating the process until the function of the entire program has been abstracted and can be compared with the specification. Functional testing was performed using boundary value, equivalence partition testing, i.e., dividing the requirements into valid and invalid equivalence classes and making up tests that check the boundaries of the classes. Structural testing: was performed to achieve 100% statement coverage, i.e., making up a set of tests to guarantee that 100% of the statements in the program have been executed.

As a blocked subject-project study, each subject used each technique and tested each program. The results were that code reading found more faults than functional testing, and functional testing found more faults than structural testing. Also, code reading found more faults per unit of time spent than either of the other two techniques.

Other conclusions from the study include the fact that the code readers were better able to assess the actual quality of the code that they read than the testers. And in fact, the structural testers were better able to assess the actual quality of the code they read than the functional testers. That

is, the code readers felt they only found about half the defects (and they were right), where the functional testers felt that had found about all the defects (and they were wrong). Also, after the completion of the study over 90% of the participants thought functional testing worked best. This was a case where their intuition was clearly wrong.

Based upon this study, reading was implemented as part of the SEL development process. However, much to our surprise, reading appeared to have very little effect on reducing defects. This lead us to two possible hypotheses:

Hypothesis 1: People did not read as well as they should have as they believed that testing would make up for their mistakes

To test this first hypothesis, we ran an experiment that showed that if you read and cannot test you do a more effective job of reading than if you read and know you can test. This supported hypothesis 1.

Hypothesis 2: There is a confusion between reading as a technique and the method in which it is embedded, e.g., inspections.

This addresses the concern that we often use a reading method (e.g., inspections or walk-throughs) but do not often have a reading technique (e.g., reading by stepwise abstraction) sufficiently defined within the method. To some extent, this might explain the success of our experiment over the ones by Hetzel and Myers.

Thus we derived the following conclusions from the studies to date:

- Reading using a particular technique is more effective and cost effective than specific testing techniques, i.e., the reading technique is important. However, different approaches may be effective for different types of defects.
- Readers needs to be motivated to read better, i.e., the reading motivation is important.
- We may need to better support the reading process, i.e., the reading technique may be different from the reading method.

2.2 The Cleanroom approach

The Cleanroom approach, as proposed by Harlan Mills [Currit, Dyer, Mills - 1986], seemed to cover a couple of these issues, so we tried a controlled experiment at the University of Maryland to study the effects of the approach.

The goal of this study was to analyze the Cleanroom process in order to evaluate and compare it to a non-Cleanroom process with respect to the effects on the process, product and developers [Selby, Basili, Baker - 1987]. This study was conducted using upper division and graduate students at the University of Maryland. The problem studied was an electronic message system of about 1500 LOC. The experimental design was a replicated project, 15 three-person teams (10 used Cleanroom). They were allowed 3 to 5 test submissions to an independent tester. We collected data on the participants' background, attitudes, on-line activities, and testing results.

The major results were:

- With regard to process, the Cleanroom developers (1) felt they more effectively applied off-line review techniques, while others focused on functional testing, (2) spent less time on-line and used fewer computer resources, and (3) tended to make all their scheduled deliveries
- With regard to the delivered product, the Cleanroom products tended to have the

following static properties: less dense complexity, higher percentage of assignment statements, more global data, more comments, and the following operational properties: the products more completely met the requirements and a higher percentage of test cases succeeded.

- With regard to the effect on the developers, most Cleanroom developers missed program execution, modified their development style, but said they would use the Cleanroom approach again.

2.3 Cleanroom in the SEL

Based upon this success, we decided to try the Cleanroom approach in the SEL [Basili and Green - 1994]. This was the basis for a case study and we used the Quality Improvement Paradigm to set up our learning process. The QIP consists of 6 steps and we define them here relative to the use of Cleanroom:

Characterize: What are the relevant models, baselines and measures? What are the existing processes? What is the standard cost, relative effort for activities, reliability? What are the high risk areas? (Figure 2)

Set goals: What are the expectations, relative to the baselines? What do we hope to learn, gain, e.g., Cleanroom with respect to changing requirements? (Figure 2)

Choose process: How should the Cleanroom process be modified and tailored relative to the environment? E.g., formal methods hard to apply, require skill; may have insufficient data to measure reliability. Allow back-out options for unit testing certain modules.

Execute: Collect and analyze data based upon the goals, making changes to the process in real time.

Analyze: Try to characterize and understand what happened relative to the goals; write lessons learned.

Package: Modify the process for future use.

There were many lessons learned during this first application of the Cleanroom approach in the SEL. However, the most relevant to reading were that the failure rate during test was reduced by 25% and productivity increased by about 30%, mostly due to fact that there was a reduction in the rework effort, i.e., 95% as opposed to 58% of the faults took less than 1 hour to fix. About 50% of code time was spent reading, as opposed to the normal 10%. All code was read by 2 developers. However, even though the developers were taught reading by stepwise abstraction for code reading, only 26% of the faults were found by both readers. This implied to us that the reading technique was not applied as effectively as it should have been, as we expected a more consistent reading result.

During this case study, problems, as specified by the users, were recorded and the process was modified.

Based upon the success of the first Cleanroom case study, we began to define new experiments with the goal of applying the reading technique more effectively. The project leader for first project became process modeler for the next two and we began to generate the evolved version of the SEL Cleanroom Process Model. Thus we moved our experimental paradigm from a case study to a multi-project analysis study. Figure 3 gives an overview of the projects studied to date. A fourth project has just been completed but the results have not yet been analyzed.

Cleanroom has been successful in the SEL. Although there is still some room for improvement in reading and abstracting code formally, a more major concern is the lack of techniques for reading various documents reading, most specifically, requirements documents. This provided our motivation for the continual evolution of reading techniques both inside and outside the Cleanroom life cycle model. Specific emphasis is on improving reading technology

for requirements and design documents.


	Sample Measures	Sample Baseline	Sample Expectation
PROCESS	Effort distribution		Increased design % due to emphasis on peer review process
	Change profile		
COST	Productivity	Historically, 26 DLOC per day	No degradation from current level
	Level of rework		
	Impact of spec changes		
RELIABILITY	Error rate	Historically, 7 errors per KDLOC	Decreased error rate
	Error distribution		
	Error source		

Figure 2: Sample Measures, Baselines, and Expectations

Technology Evaluation Steps	Off-line Reading Technology Controlled Experiment	Off-line Cleanroom Controlled Experiment	SEL Cleanroom Case Study 1	SEL Cleanroom Case Study 2	
				project 2A	project 2B
Team Size	32 individual participants	3-person development teams (10 Cleanroom teams, 5 control teams)	3-person development team, 2-person test team	4-person development team, 2-person test team	14-person development team, 4-person test team
Project Size and Application	small (145-365 LOC) sample FORTRAN programs	1500 LOC electronic message system for graduate lab course	40 KDLOC FORTRAN flight dynamics production system	22 KDLOC FORTRAN flight dynamics production system	160 KDLOC FORTRAN flight dynamics production system
Results	reading techniques appear more effective than testing techniques for fault detection	Cleanroom teams use fewer computer resources, satisfy requirements more successfully, make higher percentage of scheduled deliveries	project spends higher percentage of effort in design, uses fewer computer resources, achieves better productivity and reliability than environment baseline	project continues trend in better reliability while maintaining baseline productivity	project reliability only slightly better than baseline while productivity falls below baseline

Figure 3. Multi-Project Analysis Study of Cleanroom in the SEL

The experiments to date convinced us that reading is a *key*, if not *the* key technical activity for

verifying and validating software work products. However, there has been little research focus on the development of reading techniques, with the possible exception of reading by stepwise abstraction, as developed by Harlan Mills.

The ultimate goal here is to understand the best way to read for a particular set of conditions. That is, we are not only interested in how to develop techniques for reading such documents as requirements documents, but under what conditions are each of the techniques most effective and how might they be combined in a method such as inspections to provide a more effective reading technology for the particular problem and environment.

The idea is to provide a flexible framework for defining the reading technology so that the definer of the technology for a particular project has the appropriate information for selecting the right techniques and method characteristics. Thus, the process definition will change depending on the project characteristics. For example, if the problem and solution are well understood, we might choose a waterfall process model; if a high number of omission faults are expected, we might emphasize a traceability reading approach embedded in design inspections; when embedding a traceability reading in design inspections, we might make sure a traceability matrix exists.

As stated in the introduction, we believe there are many factors that affect the way a person reads, e.g., the reviewer's role, the reading goals, the work product. Based upon these studies, we also believe that (1) techniques can be developed that will allow us better define how we should read, and (2) using these techniques, effectively embedded in the appropriate methods, can improve the effects of reading. For example, end-users read software requirements differently than do software testers, developers read for interface defects differently than they read for missing initialization. The more I know about what kinds of defects each of the views is most effective in tracking, the better I am able to promote and manipulate that kind of reading technique in the method I am using.

We need to improve the reading of all kinds of documents and more deeply understand the relationship between techniques and methods and the dimensions of both. For example, consider the following dimensions of a reading technique:

- Input object: Requirements, specification, design, code, test plan,...
- Output object: set of anomalies
- Approach: Sequential, path analysis, stepwise abstraction, ...
- Formality: Reading, correctness demonstrations, ...
- Emphasis: Fault detection, traceability, performance, ...
- Method: Walk-throughs, inspections, reviews, ...
- Consumers: User, designer, tester, maintainer, ...
- Product qualities: Correctness, reliability, efficiency, portability,...
- Process qualities: Adherence to method, integration into process,...
- Quality view: Assurance, control, ...

We have spent some energy trying to develop and evaluate reading techniques based upon the dimension and historical data. The goal is to define a set of reading technologies that can be tailored to the document being read and the goals of the organization for that document. The technology should be usable in existing methods, such as inspections.

2.4 Scenario-Based Reading

We have defined an approach to generating a family of reading techniques. It consists of building operational scenarios based upon combining two dimensions of the technique. An operational scenario requires the reader to (1) create an abstraction of the product (based on

one dimension) (2) answer questions based on the abstraction (based on another dimension). The choice of abstraction and the types of questions asked may depend on the document being read, the problem history of the organization or the goals of the organization. The scenarios try to take advantage of the dimensions of a technique (Figure 4).

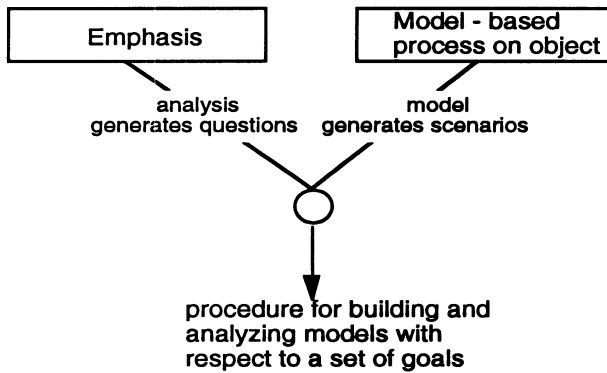


Figure 4. Building Focused Tailored Reading Techniques

Two different reading techniques, within the family, have been defined for requirements documents: defect based reading and perspective based reading

For our study, defect based reading was defined for reading SCR style documents. Defect based reading focuses on modeling different defect classes, creating three different scenarios based upon the data type consistency, safety properties, and ambiguity/missing information. The analysis questions were generated by combining/abstracting a set of checklist questions for requirements documents.

For our study, perspective based reading was defined for reading natural language requirements documents. Perspective-based reading focuses on different product customer perspectives, e.g., reading from the perspective of the software designer, the tester, or the end-user. The analysis questions were generated by focussing predominantly on various requirements type errors, e.g., incorrect fact, omission, ambiguity, and inconsistency.

To provide a little more detail into scenarios in general, and perspective based reading reading in particular, consider as an example, test-based reading.

Reading Procedure: For each requirement, make up a test or set of tests that will allow you to ensure that the implementation satisfies the requirement. Use your standard test approach and test criteria to make up the test suite. While making up your test suite for each requirement, ask yourself the following questions:

1. Do you have all the information necessary to identify the item being tested and to identify your test criteria? Can you make up reasonable test cases for each item based upon the criteria?
2. Is there another requirement for which you would generate a similar test case but would get a contradictory result?
3. Can you be sure the test you generated will yield the correct value in the correct units?
4. Are there other interpretations of this requirement that the implementor might make based upon the way the requirement is defined? Will this effect the test you made up?
5. Does the requirement make sense from what you know about the application and from

what is specified in the general description?

Each of the techniques aims at being (1) associated with the particular document (e.g., requirements) and notation (e.g., English text) in which the document is written, (2) tailorable, based upon the project and environment characteristics (3) detailed, in that it provides the reader a well-defined set of steps to follow, (4) specific, in that the reader has a particular purpose or goal for reading the document and the procedures support that goal, (5) focused, in that a particular technique provides a particular coverage of the document, and a combination of techniques provides coverage of the entire document, (6) studied empirically to determine if and when it is most effective.

Each of the techniques has been studied experimentally. The first series of experiments are aimed at discovering if scenario based reading is more effective than current practices. A second series will be used to discover under which circumstances each of the various scenario based reading techniques is most effective.

In the defect-based reading study, the goal was to analyze defect-based reading, ad-hoc reading and check-list based reading to evaluate and compare them with respect to their effect on fault detection effectiveness in the context of an inspection team from the viewpoint of quality assurance. The study was applied using graduate students at the University of Maryland. The requirements documents were written in the SCR notation. They were a water Level Monitoring System and a Cruise Control System. The experimental design is a blocked subject-project: Partial factorial design, replicated twice with a total of 48 subjects [Porter, Votta, Basili - 1995].

Major results were that (1) the defect-based readers performed better than ad hoc and checklist readers with an improvement of about 35%, (2) the defect-based reading procedures helped reviewers focus on specific fault classes but were no less effective at detecting other faults, and (3) checklist reading was no more effective than ad hoc reading.

Perspective-based reading is currently under study. The goal for the perspective-based reading evaluation was to analyze perspective-based reading, NASA's current reading technique to evaluate and compare them with respect to their effect on fault detection effectiveness in the context of an inspection team from the viewpoint of quality assurance. Two studies have been performed in the SEL environment using generic requirements documents written in English (ATM machine, Parking Garage) and NASA type functional specifications (Two ground support AGSS sub-systems). The experimental design is again a blocked subject-project using a partial factorial design. It has been applied twice, with a total of 25 subjects [Basili, Green, Laitenberger, Schull, Sorumgaard - 1995].

Preliminary indications for perspective based reading are also positive. Where there is any statistical significance, perspective-based reading appears to be more effective in uncovering defects and teams consisting of perspective based readers appear to do better than the standard reading techniques used.

3. CONCLUSION

Defect-based reading has been evaluated in experiments and has so far been shown to be superior to existing current practices. Perspective-based reading is being evaluated in experiments and the results so far appear promising.

Specifically we have run the experimental gamut from blocked subject-project experiments (reading vs. testing) to replicated projects (University of Maryland Cleanroom study) to a cased study (the first SEL Cleanroom study) to multi-project variation (the set of SEL Cleanroom projects) and now back to blocked subject project experiments (for scenario based reading). See Figure 5.

Scopes of Evaluation			
		#Projects	
		One	More than one
# of Teams	One	3. Cleanroom (SEL Project 1)	4. Cleanroom (SEL Projects, 2,3,4, ...)
per Project	More than one	2. Cleanroom at Maryland	1. Reading vs. Testing 5. Scenario Reading vs. ...

Figure 5. Series of Studies

In the future, we plan to replicate these experiments in many different environments. Various groups at different sites are already replicating some of the earlier experiments. Most of these are members of ISERN, the International Software Engineering Research Network, whose goal is specifically to perform and share the results of empirical studies.

We will continue to develop operational scenario reading techniques (e.g., design reading, etc.) and test their effectiveness in experiments. Future work also includes the consideration of tool support for the technologies developed.

4. REFERENCES

Basili, V.R. (1985) Quantitative Evaluation of Software Methodology, Keynote Address, *First Pan Pacific Computer Conference*, Melbourne, Australia.

Basili, V.R. (1989) Software Development: A Paradigm for the Future, *COMPSAC '89*, Orlando, Florida, pp.471-485.

Basili, V.R., Caldiera, G., McGarry, F., Pajersky, R., Page, G., Waligora, S. (1992) The Software Engineering Laboratory--An Operational Software Experience Factory, *14th International Conference on Software Engineering*, Melbourne, Australia.

Basili, V.R. and Green, S. (1994) Software Process Evolution at the SEL, *IEEE Software*, pp 58-66.

Basili, V.R., Green, S., Laitenberger, O.U., Schull, F. and Sorumgaard, S. (1995) To be published) The Empirical Investigation of Perspective-Based Reading (in progress).

Basili, V.R. and Rombach, H.D. (1988) The TAME Project: Towards Improvement-Oriented Software Environments, *IEEE Transactions on Software Engineering*, vol.14, no.6.

Basili, V.R. and Selby, R. (1987) Comparing the Effectiveness of Software Testing Strategies, *IEEE Transactions on Software Engineering*, pp.1278-1296.

- Basili, V.R. and Weiss, D.M. (1984) A Methodology for Collecting Valid Software Engineering Data, *IEEE Transactions on Software Engineering*, pp.728-738.
- Currit, P.A., Dyer, M. and Mills, H.D. (1986) Certifying the Reliability of Software, *IEEE Transactions on Software Engineering*, vol.SE-12, pp. 3-11.
- Hetzel, W.C. (1972) An Experimental Analysis of Program Verification Problem Solving Capabilities as They Relate to Programmer Efficiency, *Computer Personnel*, vol.3, pp.10-15.
- Linger, R.C., Mills, H.D. and Witt, B.I. (1979) *Structured Programming: Theory and practice*, Reading, MA: Addison-Wesley.
- Myers, G.J. (1978) A Controlled Experiment in Program Testing and Code Walkthroughs Inspections, *Communications ACM*, pp.760-768.
- Porter, A.A., Votta, L.G. and Basili, V.R. (1995) Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment, *IEEE Transactions on Software Engineering*, vol.21, no.6, pp.563-575.
- Selby, R., Basili, V.R. and Baker, T. (1987) Cleanroom Software Development: An Empirical Evaluation, *IEEE Transactions on Software Engineering*, pp 1027-1037.