# Replicated Studies:
# Building a Body of Knowledge about Software Reading Techniques

Forrest Shull[1], Jeffrey Carver[2], Guilherme H. Travassos[3] ,
José Carlos Maldonado[4], Reidar Conradi[5], and Victor R. Basili[6]

*[1]Fraunhofer Center—Maryland*
*USA*
***fshull@fc-md.umd.edu***

*[2]University of Maryland, College Park, Dept. of Computer Science*
*USA*
***carver@cs.umd.edu***

*[3]Federal University of Rio de Janeiro,*
*COPPE-Systems Engineering and Computer Science Program*
*Brazil*
***ght@cos.ufrj.br***

*[4]University of São Paulo at São Carlos, Dept. of Computer Science*
*Brazil*
***jcmaldon@icmsc.sc.usp.br***

*[5]Norwegian University of Science and Technology*
*Norway*
***conradi@idi.ntnu.no***

*[6]Fraunhofer Center—Maryland and University of Maryland, College Park,*
*Dept. of Computer Science*
*USA*
***basili@cs.umd.edu***

## Introduction

In Software Engineering, researchers are continually developing new tools, techniques and methods. The problem is that very often these new technologies never make it out of the research laboratory into real-world practice, and when they do, there is often little empirical data capable of showing their likely effect in practice. Therefore, software developers have a plethora of development technologies from which to choose, but often little guidance for making the decision. Researchers and developers can both benefit from a better understanding of the practical effects and implications of new technologies. Such an understanding will allow decisions to be made not based on anecdote, hearsay, or hype, but rather based on solid empirical data. Many software engineers are still surprised to learn that 25 years of empirical research activities in software engineering have yielded important insights that can aid in the decision-making process.

Both researchers and practitioners have a general need for properly evaluated technologies that are well understood, but their specific needs and goals are slightly different. Researchers need to perform focused evolution of their own technologies in the lab to understand when further development and assessment are required or when the technology is ready for deployment to practitioners. On the other hand, developers need reliable support to help determine which technologies to select for best use in their environment.

An important way in which information can be built up about a technology is by running empirical studies. While a single well-run study can provide some specific information about the technology within a particular context, the results of any single study on almost any process depend to a large degree on a large number of relevant context variables. Thus, the results of any single study cannot be assumed *a priori* to apply in another context. Obtaining more general information about a technology requires the running of multiple studies under different contexts. Multiple studies allow the specific results of a single study to be validated and/or generalized across varying environments. However, some kind of approach is necessary to abstract the specific results from multiple studies into a useful and reliable body of knowledge capable of providing general recommendations about a technology.

Based on its usefulness in other fields, meta-analysis, which provides a statistical basis for drawing conclusions across multiple studies, could be a promising vehicle for use in software engineering research. However, initial attempts to apply it to studies of software development technologies have not been successful [Miller00], perhaps reflecting the level of maturity of the field. To apply meta-analysis, the field must be at a level of sophistication where different researchers are able to agree upon a common framework for planning, running, and reporting studies. While we believe that increased experience with running empirical studies will eventually lead to guidelines to facilitate the combining of software studies, in the meantime we must use less formal methods to combine results until the field reaches this level of sophistication.

We advocate a more informal approach to building up a body of knowledge while the field matures to the point of using meta-analysis. In this approach, replication is used to run families of studies that are designed *a priori* to be related. Because new studies are based upon the designs of existing ones, it becomes easier to identify the context variables that change from one study to another. By comparing and contrasting the results of studies in the same family, researchers can reason about which context variables have changed and hypothesize what their likely effects on the outcome have been. As more studies become part of the family, hypotheses can be refined, or supported with more confidence by additional data. By using this informal approach, we can work toward producing a robust description of a technology's effects, specifying hypotheses at varying levels of confidence.

Informal approaches such as this have been used before in software engineering research, for example, to formulate, successively refine, and then test hypotheses concerning effective Object-Oriented development [Wood99]. Our work is directly in line with the multi-method approach advocated by Wood *et al.*

Later in this chapter, we will present a more detailed discussion of various types of replications and why they are necessary for allowing the variation of important factors in a controlled way to study their effects on the technology. For clarity a brief working definition of an replication will be given here. While in many contexts, the term *replication* implies repeating a study without making any changes, this definition is too narrow for our purposes. In this work, a replication will be a study that is run, based on the results and design of previous study, whose goal is to either verify or broaden the applicability of the results of the initial study. For example, the type of replication where the same exact study is run could be used to verify results of an original study. On the other hand, if a researcher wished to explore the applicability of the results in a different context, then the design of the original study may be slightly modified but still considered a replication.

This chapter will provide an example of this informal approach and how it has been used in the evolution of two sets of software reading techniques. We will present a brief description of each set of replications, focusing on the lessons learned about the reading technology based on the results of the original study and the replication together. We will also discuss what we learned about the technologies from the entire series of studies, as well as what we learned about reading techniques in general. We will indicate which of these lessons were due directly to the process of replication and could not have been learned through a single study.

We will conclude this chapter with some lessons learned about replicating studies. Based on the replications discussed as examples, we will discuss what we learned about making replications more effective.

## Reading Techniques

To illustrate what we mean about a body of knowledge about a particular software development technology, we give the example in this paper of recent work in the area of “software reading techniques”. This section gives some background on the technology and how it relates to the larger set of software development approaches.

### *Reading Techniques in General*

A reading technique can be defined as “a series of steps for the individual analysis of a textual software product to achieve the understanding needed for a particular task” [Shull02a]. This definition has three main parts. First, the series of steps gives the reader guidance on how to achieve the goal for the technique. By defining a concrete set of steps, we give all readers a common process to work from which we can later improve based on experience. In contrast, in an ad hoc, or unstructured, reading process, the reader is not given direction on how to read, and readers use their own processes. Without a standardized process, improvement of the process is much more difficult. Secondly, a reading technique is for individual analysis, meaning that the aim of the technique is to support the understanding process within an individual reader. Finally, the techniques strive to give the reader the understanding that they need for a particular task, meaning that the reading techniques have a particular goal and they try to produce a certain level of understanding related to that goal [Shull98].

The “series of steps” that each reviewer receives consists of two major components: A concrete **procedure** that can be followed to focus on only the information in the review document that is important for the quality aspects of interest, and **questions** that explicitly ask the reader to think about the information just uncovered in order to find defects.

Using the broad description outline above, different families of reading techniques can be instantiated for many different purposes [Basili96]. For example, a candidate Object-Oriented framework[1] could be evaluated for reuse by means of a reading technique. One option would be for a textual

[1] Here we use the term “framework” according to the specific technical definition of a particular system designed for reuse: an object-oriented class hierarchy augmented with a built-in model that defines how the objects derived from the hierarchy interact with one another to implement some functionality. A framework is tailored to solve a particular problem by customizing its abstract and concrete classes, allowing the framework architecture to be reused by all specific solutions within a problem domain. By providing both design and infrastructure for developing applications, the framework approach promises to develop applications faster [Lewis95].

representation of the framework to be reviewed by an individual from the point of view of whether the functionality supported would be useful for the planned project. A reading technique could be developed for this task to give the

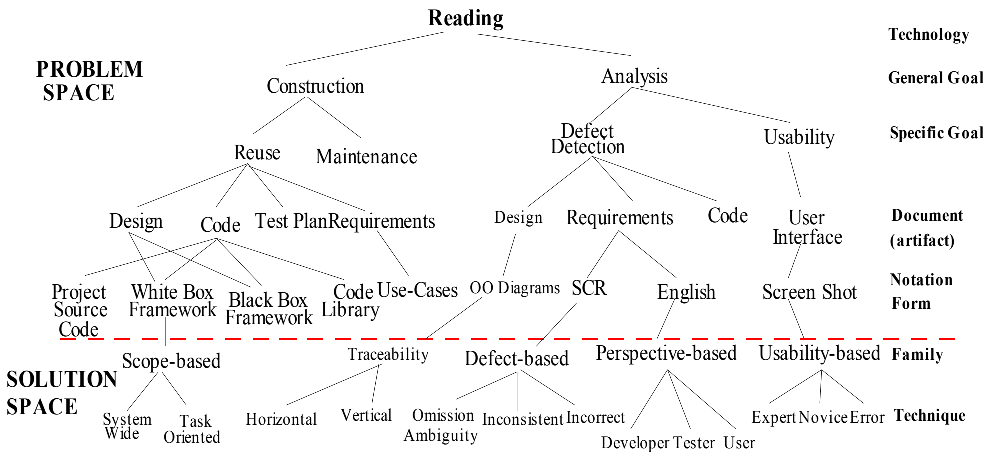

Figure 1 – Families of Reading Techniques

reviewer a procedure that could be followed to understand what types of functional descriptions and interface issues to focus on. Sets of questions would be used to make the reviewer consider important quality aspects that would affect reuse, such as how the expectations for flow of control and interface parameters of the reusable framework match the expectations for the rest of the system.

The taxonomy of reading technique families developed to date is shown in Figure 1. The upper part of the tree (over the horizontal dashed line) models the problems that can be addressed by reading. Each level represents a further specialization of a particular software development task according to classification attributes that are shown in the rightmost column of the figure. The lower part of the tree (below the horizontal dashed line) models the specific solutions we have provided to date for the particular problems represented by each path down the tree. Each family of techniques is associated with a particular goal, artifact, and notation.

This tailorablity is an important attribute of reading techniques, by which we mean that each reading technique defined is specific to a given artifact and to a goal. For example, one specific goal could be defect detection. Software reading is an especially useful method for detecting defects since it can be performed on all documents associated with the software process, and can be applied as soon as the documents are written. Given this goal, we could imagine software reading techniques tailored to natural language requirements documents, since requirements defects (omission of information, incorrect facts, inconsistencies, ambiguities, and extraneous information) can directly affect the quality of, and effort required for, the design of a system. For this technique, the procedure would be concerned with understanding what information in the requirements is important to verify (namely, that information which is required by downstream users of the requirements document to build the system). Questions could be concerned with verifying that information to find defects that may not be uncovered by a casual or unstructured reading.

Reading techniques for the purpose of defect detection, especially as used to augment software inspections, has been one of the most widely applied branches of the tree in Figure 1. For this reason, this subset of the technology is described in more detail in the next section.

## *Reading Techniques for Defect Detection*

A software inspection aims to guarantee through defect detection and removal that a particular software artifact is complete, consistent, unambiguous, and correct enough to effectively support further system development. For instance, inspections have been used to improve the quality of a system's design and code [Fagan76]. Typically, inspections require individuals to review a particular artifact, then to meet as a team to discuss and record defects, which are sent to the document's author to be corrected. Most publications concerning software inspections have concentrated on improving the inspection meetings while assuming that individual reviewers are able to effectively detect defects in software documents on their own (e.g. [Fagan86], [Gilb93]). However, empirical evidence has questioned the importance of team meetings by showing that meetings do not contribute to finding a significant number of new defects that were not already found by individual reviewers [Porter95, Votta93].

Software reading techniques can be applied to improving the effectiveness of software inspections by improving the effectiveness of individual reviewers, during their preparation activities before the inspection meeting. Reading techniques provide procedural guidelines containing tested procedures for effective individual inspection—a step that is often ignored in state-of-the-practice inspection processes.

Reading techniques combine and emphasize three "best practices" that we have found helpful based on personal experience for effective inspections in a variety of contexts. While any of the three practices can be useful in and of itself, their integration in a unified inspection approach has been shown to be particularly valuable. Those practices are:

1. *Giving each reviewer a particular and unique focus (or perspective) on the document under review.* Studies such as [Basili96] and personal experience have shown that reviewers work better when they have a clear focus than when they feel themselves

responsible for all types of defects, in all parts of the document. Additionally, having a unique focus means that each reviewer has clear responsibility for a certain aspect of the document and can't count on another reviewer catching any missed defects.

2. *Making individual review of a document an active (rather than passive) undertaking.* Reviewers tend to make a more thorough review when they are actively engaged in working with the information contained in a document than when they can get by with merely reading it over. This has been an important principle driving the development of certain inspection approaches, such as Active Design Reviews [Knight93].
3. *Articulating a taxonomy of the types of defects of interest, and giving the reviewer an understanding how to look for those types of issues during the individual review.* Reviewers do a better job of reviewing documents when they have a good idea what they are looking for. In tailoring reading techniques to specific project environments, defect taxonomies must be made explicit and reflected in the questions that are given to the reviewer.

Two families of reading techniques that have received the most effort in terms of training and evaluation include Perspective-Based Reading (PBR) for requirements inspections, and Object-Oriented Reading Techniques (OORTs) for inspection of high-level UML designs.

## PBR

Perspective-Based Reading (PBR) is a family of techniques that have been specifically designed for the review of English-language requirements. In planning a PBR review, the potential stakeholders of the requirements document are identified and the differing perspectives are used to give different reviewers a particular and unique quality focus: Each reviewer is asked to take the perspective of some downstream user of the requirements, and be concerned with only that quality focus during the review. Some examples of requirements stakeholders could include:

- a user, who has to validate that the requirements contain the right set of functionality for his/her needs;
- a designer, who has to verify that the requirements will allow a correct design to be created;
- a tester, who must ensure that the functionality is specified in such a way as to be testable in the final system.

Review is made into an active undertaking by asking each reviewer to create a high-level version of the work products that the appropriate requirements stakeholder would have to create as part of his or her normal work activities. In this way, the reviewer is forced to manipulate the information in the document in a way that approximates the actual work activities that document must be able

| ***Defect*** | ***Applied to requirements*** | ***Applied to design*** |
|---|---|---|
| *Omission* | (1) some significant requirement related to functionality, performance, design constraints, attributes or external interface is not included; (2) responses of the software to all realizable classes of input data in all realizable classes of situations is not defined; (3) missing sections of the requirements document; (4) missing labeling and referencing of figures, tables, and diagrams; (5) missing definition of terms and units of measures [ANSI84]. | One or more design diagrams that should contain some concept from the general requirements or from the requirements document do not contain a representation for that concept. |
| *Incorrect Fact* | A requirement asserts a fact that cannot be true under the conditions specified for the system. | A design diagram contains a misrepresentation of a concept described in the general requirements or requirements document. |
| *Inconsistency* | Two or more requirements are in conflict with one another. | A representation of a concept in one design diagram disagrees with a representation of the same concept in either the same or another design diagram. |
| *Ambiguity* | A requirement has multiple interpretations due to multiple terms for the same characteristic, or multiple meanings of a term in a particular context. | A representation of a concept in the design is unclear, and could cause a user of the document (developer, low-level designer, etc.) to misinterpret or misunderstand the meaning of the concept. |
| *Extraneous Information* | Information is provided that is not needed or used. | The design includes information that, while perhaps true, does not apply to this domain and should not be included in the design |

Table 1 - Types of software defects, with specific definitions for requirements and design

to support. Additionally, the intermediate artifacts to be created during the review should be chosen carefully for their reusability downstream. The objective is not to duplicate work done at other points of the software development process, but to create *representations* that can be used as a basis for the later creation of more specific work products and that can reveal how well the requirements can support the necessary tasks. For the designer, tester, and user perspectives discussed above, the relevant work products would be a high-level design of the system described by the requirements, a test plan for the system, and an enumeration of the functionality described in the requirements, respectively.

Finally, questions are distributed at key points of the procedure to focus reviewers on an explicit defect taxonomy. As the reviewer goes through the steps of constructing the intermediate artifact, he or she is asked to answer a series of questions about the work being done. There is at least one question for every applicable type of defect. When the requirements do not provide enough information to answer the questions, this is usually a good indication that they do not provide enough information to support the user of the requirements, either. This situation should lead to one or more defects being reported so that they can be fixed before the requirements need to be used to support that particular stakeholder later in the product lifecycle. As always, the defect types of interest to a project vary widely from one environment to another. However, a set of generic defect types, with associated definitions tailored to requirements inspections, is shown in Table 1 as a starting point for project-specific tailoring.

## OORTs

Object-Oriented Reading Techniques (OORTs) are a family of techniques created for review of UML designs. The OORT family of techniques has been evolving over a long series of studies in software engineering courses where inspections are being taught, and are now at the point where they have been first used by some industrial companies. Like PBR, the OORTs drive the inspection of a design by means of key perspectives: validating that the design created is sufficient for implementing the desired functionality that was set forth in the requirements (called vertical reading), and verifying that the design documents themselves are consistent enough to support detailed design and eventual implementation (known as horizontal reading). However, because UML describes a document standard that is not implemented in all environments in exactly the same way, the OORTs have been modularized. There are seven techniques altogether, one for each comparison between two (or in some cases three) UML diagrams that can effectively be compared. In this way, tailoring to project environments can be done more easily, as projects can simply choose the subset of OORTs that are appropriate for the subset of UML diagrams that they are using in their environment. The complete set of OORTs is defined as shown in Figure 2. Each line between the software artifacts represents a reading technique that has been defined to read one against the other.

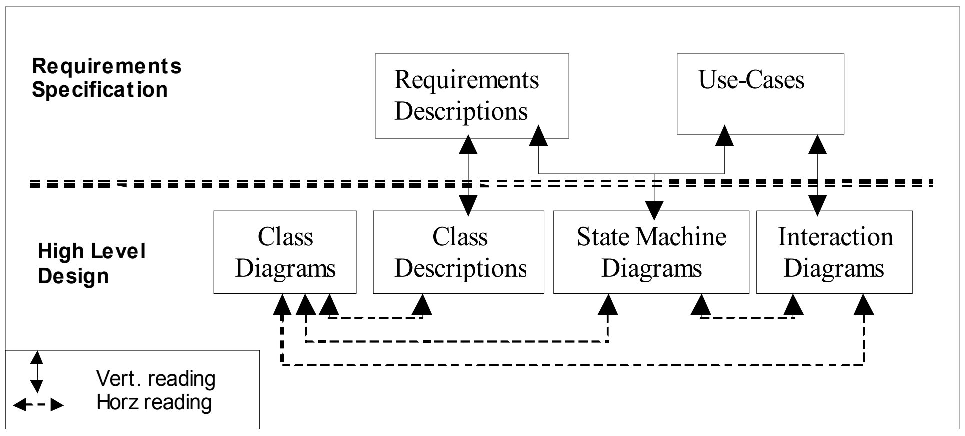

Figure 2 - The set of OORTs (each arrow represents one technique) that has been defined for various design artifacts.

OORT review is an active undertaking because effort must be expended to reconcile the different views of the same information contained in different UML diagrams being compared. The OORT procedures guide the reviewer to walk through the various diagrams, marking related information in each. For example, sequence diagrams show many specific messages being passed between objects, which taken in the aggregate might correspond to a particular high-level use case. The OORT reviewer has to understand which sequence of messages corresponds to the higher-level functional description, and mark both documents to show that the overall concept is the same in both cases. Once the equivalent information has been marked on various documents, then the reviewer can analyze whether it is represented correctly in both cases.

This analysis is guided by means of specific questions included in the techniques. As in PBR, the list of questions is tied directly to the list of defect types explicitly identified in the taxonomy. An initial set of UML defect types is included in Table 1, shown alongside the requirements definitions to show how generic defect types can be redefined in various phases of the lifecycle.

## Building a Body of Knowledge

As mentioned in the Introduction, the sophistication level of Empirical Software Engineering is not yet at the level where families of studies are organized with the goal of drawing broad conclusions. On the other hand, because of the relatively limited scope within which a specific set of results is applicable, we need to begin to aggregate results of multiple studies to move the field to this higher level of sophistication. Therefore, we need to develop a way to organize and plan the studies in order to get the most benefit from the combination of their results. This means that the studies should be planned so that the important context variables can be varied in some controlled way in order to study their effects on the results of using the technology.

The tree in Figure 1 illustrates a way that results of different studies can be abstracted up to draw lessons learned about specific technologies or classes of technologies. While it is not realistic to assume the field will ever be at the point where all of the studies run on technologies have been coordinated and planned with the idea of abstraction of results in mind, we think that the hierarchy in Figure 1 provides a good model to start from to reason about how some such studies may fit together. As we do have studies that fit into the tree, we can start with those studies and abstract the results to show the value of putting studies together. If the aggregated results of studies, which were not specifically planned with the goal of result abstraction in mind, can be abstracted successfully, then there is promise that a set of studies that are planned with this goal in mind can provide even better results.

### *Abstraction across a series of studies*

If we can plan a series of studies on a given technology such that the studies are coordinated to address different context variables, we will be better able to draw conclusions about the technology. Each of the various context variables (e.g. the process experience of the developers using the technology, the problem domain in which it is applied, or the level of procedural constraint given to the subjects) is a dimension along which the design of a study can be altered. That is, each context variable represents a specific characterization of the environment that can be modified for each study. To rigorously test the effects of different context variables on results, it would be necessary to run further studies that made different decisions for the relevant variable. If this series of studies can be done systematically, then a body of knowledge can be built up that represents a real understanding of some aspect of software engineering. A body of knowledge supports decision-making because the results of the individual studies can be viewed together and abstracted to give deeper insight into the technology being studied. For example:

- If a new environment is substantially similar to the environment of a previous study that helped build the body of knowledge, the previous results about the use of the technology can be directly useful;
- If the new environment is significantly different from others that were used to build up the body of knowledge, the aggregate data can still be useful by observing across a number of different environments which context variables have an impact on effectiveness and which not.

In this way, articulating the important context variables can serve as a means of organizing a set of related empirical studies.

Thus, families of related studies are the cornerstone to building knowledge in an incremental manner. However, performing such studies in an ad hoc way (without careful planning of relationships within the family) is inadequate. For example, an attempted meta-analysis of several studies on software inspections was unable to come to any higher-level conclusions because the studies were not designed, run, or reported in ways that were compatible [Miller00]. What is necessary is a framework for planning studies and data analysis in a way that enables reasoning about the commonalties of the results.

The key method for creating families of related studies is through the use of replications. The main idea behind a replication is to take a study run to investigate some technology and repeat that study. The replication normally makes some slight modifications to the original study in order to either test new hypotheses or to verify existing results. In this way, a body of knowledge is built through studies designed to relate to one another, rather than through trying to impose a framework post-hoc on unrelated studies.

### Types of Replications

A taxonomy describing how replications can relate to the original study was proposed in the TSE special issue on empirical studies [Basili99]. In this view there were three major categories of replications based on how key dimensions were varied. Each of the dimensions below illustrates a particular reason for running a replicated study, although it should be noted that replications often contain several different types of changes from the original (for example, a replication might be primarily intended to study some particular change to the technology itself but also might introduce some refinements into the study design at the same time).

I. **To duplicate as accurately as possible the original study.**
These replications are necessary to increase confidence in the validity of the study. They demonstrate that the results from the original study are repeatable and have been reported accurately. They are also useful for teaching new empirical researchers how to run studies.

An example is the ISERN Benchmark Re-Inspection Experiment (BRIE), an experiment to investigate the effectiveness of inspections and meetings that has been packaged at http://csdl.ics.hawaii.edu/techreports/96-13/96-13.html. BRIE has been designed so that it is easy to carry out, and the expected results (or at least their general characteristics) are well-known and stable across replications. Thus, experimenters in multiple environments can run the BRIE experiment and compare their results in order to calibrate their experimental methods and gain experience in running experiments.

II. **To vary *how* the particular development technology is studied.** These studies seek to increase our confidence in the results concerning a specified technology by addressing the same problem as previous studies, but altering the details of how the study is run so that threats to validity can be addressed.

**A. To address *external validity*.**
These replications use the same study design as the original but use a different type of sample population to address concerns about whether the original results can be extrapolated beyond the original subjects. Such replications can be used to investigate whether results transfer between different industrial environments, or from a laboratory environment to a particular industrial environment.

For example, an approach used at NASA's Software Engineering Laboratory was to run small focused studies first with students, to determine the feasibility of an approach or general principles. If those results proved successful then studies could be run with (more expensive) NASA personnel to test whether the results could transfer successfully into industrial practice [Basili97].

**B. To address *internal validity*.**
These replications investigate similar hypotheses to the original study but use a different design to address threats to validity resulting from the way the study is run. The goal is for the replicated study to contain different threats to validity than the original study so that, while neither is perfect on its own, both studies contribute to raising confidence that the results are rigorous and independent of the study methodology used.

For example, a study by Ciolkowski *et al.* [Ciolkowski97] replicated an earlier study of inspection techniques in which the effectiveness of individuals during a review was studied. In the earlier study, the effectiveness of review meetings was not studied directly, but was instead simulated from the data collected on individuals using statistical techniques. The replicated study collected data from both individuals and review meetings, allowing the accuracy of those statistical methods to be verified.

III. **To vary *what* is studied.**
These replications vary details of the particular development technology under study.

A. **To vary details of the technology, for *improvement* within an environment.** These replications investigate what aspects of the technology are important by systematically varying intrinsic properties of the technology and examining the results.

For example, a series of studies was undertaken at the University of Maryland to evolve a new inspection technique in an incremental matter. First the idea was shown to be feasible, then additional studies were undertaken to optimize the inspection procedure. Procedural steps were re-ordered or dropped and terminology was refined in order to better match the procedure to the work practices of the students [Shull01].

B. **To vary details of the technology, for *tailoring* to a new environment.** These replications vary certain environmentally dependent parts of the technology to identify potentially important environmental factors that can affect the results of the process under investigation, and hence the technology's suitability for various environments. Results also demonstrate the tailoring needed for various environments.

This type of replication requires the technology and other artifacts from the original study to be supplied in sufficient detail that changes can be made. This implies that the rationales for the design decisions must be provided (so that replicators know what features are easy to change and which are intrinsic to the effectiveness of the technology) along with the finished product.

For example, an experiment undertaken in 2001 in a class at the Federal University of Rio de Janeiro used a set of English-language inspection techniques that had been shown to be useful in a similar classroom environment in the United States, and translated them into Portuguese. The recorded design rationales were used to determine where key concepts needed to be translated "as is" and where local equivalents were free to be substituted.

More loosely defined, there are also replications that adapt only the basic idea behind a particular technology to a new goal or new situation. For example, an inspection approach that had proven effective for requirements inspections was adopted for use in usability inspections of web pages [Zhang99]. Key concepts were retained, such as having reviewers take the perspectives of different stakeholders of the inspected product, but adapted to the new domain, for example by focusing on expert and novice users of the web-based system rather than downstream users of the requirements

## Benefits and Risks of Replications

Running replications involves potential benefits but also dangers, which must be kept in mind while planning and running the studies.

Benefits fall into two main categories: The first benefit is that by using replications the results of a study can be validated or corrected, in the case where the original study has methodological errors. If a study is replicated and produces similar results then the confidence in the validity of those results increases. As more replications produce similar results, the level of confidence in those results increases. On the other hand, if dissimilar results are produced, further analysis may trace the discrepancy back to issues in either the original or replication, helping to debug the experimental protocols. The second benefit is that the interaction of different values in the study can be better understood. By replicating a study and changing the dependent and independent variables, the scope effect of those variables on the process under study can be better understood.

On the other hand, the dangers that can arise when replicating studies have to be taken into account. The first and most important danger is that if the replication is done poorly or by an inexperienced researcher, the results could be at best incomparable or at worst contradictory. A well run study producing contradictory results provides enough information to understand the results and how they relate to the existing body of knowledge, but a poorly run study with contradictory results can only confuse the effort to provide useful decision support, by introducing unsubstantiated data into the body of knowledge. The second danger is that in the process of trying to make the replication more interesting, the researcher might change too many variables in the study design. When this situation occurs, the results are often incomparable because there are too many potential sources of variation that might account for any differences in results. For replications, as for individual studies, the goal must be to minimize the number of rival hypotheses that can be put forward to provide an explanation of the results.

## Building a Body of Knowledge about Reading Techniques

In this section, we illustrate how the body of knowledge is being built on the subject of *software reading techniques* (described in an earlier section) by describing four of the latest replications done on this topic. In each case, we describe briefly the mechanics of running the replication (to explore some of the important issues involved in replications in general) and concentrate on what each replication contributes to the body of knowledge.

The studies described evaluated two different families of reading techniques (PBR for requirements and OORTs for high-level object oriented designs), allowing us to discuss lessons that were learned both about the specific technologies and about reading techniques in general. In this way we aim to show that replications are useful both for addressing specific practical questions about technology use and for abstracting up basic principles concerning a technology that can be used to tailor it for additional environments. These particular studies are also useful for illustrating the four types of replications described in the previous section.

Figure 3 illustrates the time frame in which each study/replication pair was run, and shows that the example replications were selected from different periods in the maturity of the techniques. Some replications followed their original studies closely in time, while others came much later.

Later in the chapter, we abstract some lessons learned about running replications in general and describe some guidelines for “lab packages” that collect and organize the information necessary for effective replications that take these lessons into account. We also describe prototype repositories of studies that are being built to instantiate these packages.

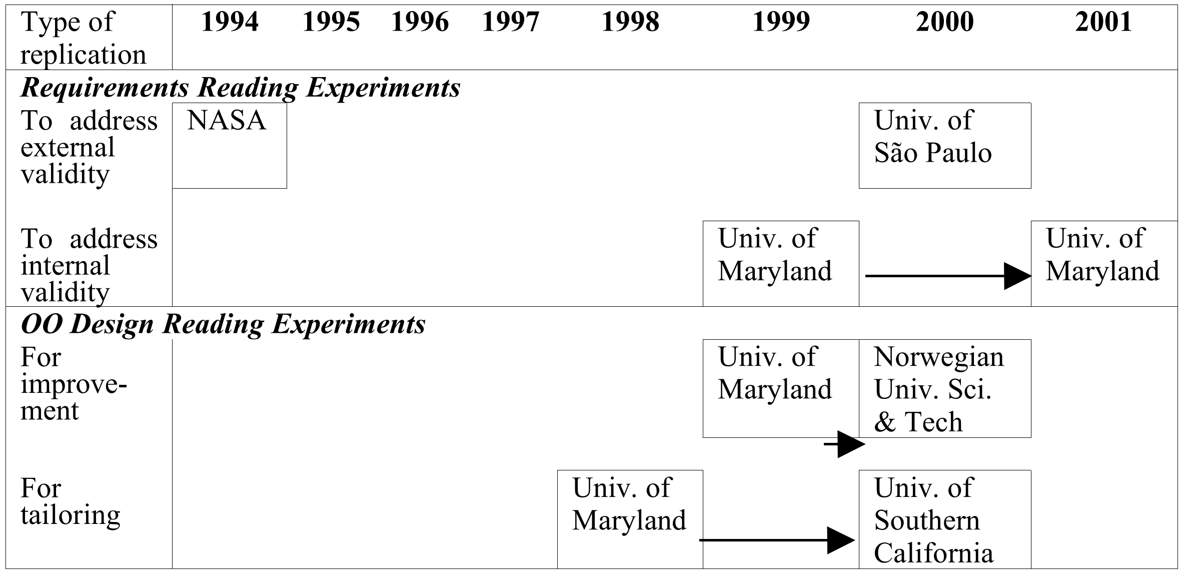
Type of replication
1994
1995
1996
1997
1998
1999
2000
2001
Requirements Reading Experiments
To address external validity
NASA
Univ. of São Paulo
To address internal validity
Univ. of Maryland
Univ. of Maryland
OO Design Reading Experiments
For improvement
Univ. of Maryland
Norwegian Univ. Sci. & Tech
For tailoring
Univ. of Maryland
Univ. of Southern California

Figure 3 – Relationships between replicated studies and originals.

### *PBR*

PBR was piloted at NASA's Goddard Space Flight Center [Basili96] in Maryland and has been adapted for industrial use and training at Allianz Life Insurance, Bosch Telecom, and Robert Bosch GmbH, Germany. In addition, a series of empirical studies of PBR with students and professionals through universities in several countries has shown that it is more effective than less procedural approaches, e.g. [Ciolkowski97, Sørumgård97, Shull98].

#### Study at University of São Paulo

In 2000 a study was run at the University of São Paulo in Brazil (USP) [Shull02b], which replicated a study that was originally run at NASA [Basili96] in order **to address the external validity**.

The study was a controlled experiment of the Perspective-Based Reading (PBR) techniques for requirements review [Basili96], designed so that it can be run in many different environments, allowing the improvement due to PBR to be studied in comparison to many other inspection approaches already in use.

Due to an experience package that was made available on the web specifically for facilitating replications[2], this experiment has been replicated many times, in different contexts.

The replicating researchers reused the same design and materials as the original study, but changed the sample population in order to assess whether the conclusions of the original study held outside of the original subjects. This context was a good choice for this replication, because many of the previous runs of this study had shown an increase in inspection effectiveness for subjects with a mid-range of experience (i.e. subjects with a basic knowledge of some requirements stakeholder role, although not experts in inspections) due to the use of PBR. As a result, the experimenters reasoned that this could be a promising technique for instructing software engineering students, as the procedural approach can be used to provide guidance when subjects have not had the experience yet to develop their own inspection processes.

The replication was undertaken by independent researchers with the support of the original experimenters, who were consulted during the design and preparation of the replication. A pilot study was run in the local environment before the main study, to increase the researchers' expertise in the technology and debug the experimental protocol.

[2] http://www.cs.umd.edu/projects/SoftEng/ESEG/manual/pbr_package/manual.html

### *Subjects*

18 undergraduate students from the University of São Paulo participated. Students had previously received introductory training in the perspectives they were asked to take during the review. The experience level of subjects was very different in the original study, which used 25 professional software developers from the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) Software Engineering Laboratory (SEL).

### *Procedure*

The central research question was to evaluate the effectiveness of a systematic (i.e. procedure-based) approach to inspections. To achieve this, the experimental design compared PBR, which provided a systematic process focused on certain classes of defects, to a nonsystematic approach that could be used for the same defect classes. The nonsystematic approach was provided by a checklist-based approach focused on the same defect taxonomy as was used to create the PBR procedures.

The experimental design consisted of training the subjects in the checklist approach and allowing them to review a document, then training them in PBR and allowing them to review a second document. Improvement was measured mainly by evaluating the *defect detection effectiveness* of each review (the average percentage of defects found). Because it was assumed that reviewers could not avoid incorporating elements of the systematic process training in the nonsystematic process, the order of applying the two review approaches was not varied (that is, systematic training always had to come after the nonsystematic review). However, two different documents were used in each review to ensure that the results were not document-specific. (These documents were from "generic" domains with which most subjects were expected to be familiar, an ATM system and a control system for a parking garage (PGCS)).

Subjects in the original experiment, being professional developers at NASA's Goddard Space Flight Center, already had their own nonsystematic procedure for inspections [SEL92]. They also had additional experimental treatments where they applied both techniques to typical documents from their own environment (called NASA_A and NASA_B). Because subjects in the replication were students who did not have existing review procedures or work documents in their environments, the reviews of NASA-specific documents were simply dropped and training in a checklist technique had to be added (as illustrated in Figure 4).

### *Data Collection*

The defect lists resulting from each subject's reviews were evaluated to determine the defect detection rate, i.e. the percentage of the seeded defects that was found by each reviewer. The data from the individual reviews was later used to investigate the defect detection effectiveness that can result from review teams by using simulation techniques to evaluate the coverage that would result when individuals were grouped. The percentage of defect occurrences found by each approach was also measured, i.e. percentage of the number of defects that would have been reported if all reviewers had been 100% effective.

Subjects were asked to fill out a background questionnaire at the beginning of the experiment, and an opinion questionnaire at the end.

*Lessons Learned*

| | **Original Study** | | **Replicated Study** | |
|---|---|---|---|---|
| **Activity** | *Group A (half of subjects)* | *Group B (half of subjects)* | *Group A (half of subjects)* | *Group A (half of subjects)* |
| **Training in checklist review** | X | X | | |
| **Review generic document** | (ATM) | (PGCS) | (ATM) | (PGCS) |
| **Review NASA document** | (NASA_A) | (NASA_B) | X | X |
| **Training in PBR review** | | | | |
| **Review NASA document, with PBR** | (NASA_B) | (NASA_A) | X | X |
| **Review generic document, with PBR** | (PGCS) | (ATM) | (PGCS) | (ATM) |

Figure 4: Order of activities in the study design. (Activities that did not occur in both the original and replicated studies are marked with an X and shaded in the appropriate place.)

Previous studies had shown that PBR was at least as good as, and sometimes better than, the subjects' usual approach. Moderately-experienced reviewers exhibited some of the largest improvements in review effectiveness when using PBR. Although not a statistically significant trend, this had indicated that less experienced reviewers might be more likely to benefit from procedural guidelines.

The São Paulo replication provided more data corroborating this supposition. For one document (ATM), the PBR group had a statistically better performance measured both by the percentage of defects that were found by the group overall as well as the total number of defect reports by unique individuals, than the group that applied the checklist. For the other document (PGCS), the effectiveness and efficiency of the group applying PBR were only slightly better.

As in previous studies, each unique stakeholder perspective found certain unique defects that were found by no other perspective. This indicates that the multiple perspectives each contribute value and really do provide a unique viewpoint of the document, minimizing overlap in effort.

Taking these results in combination with the results of the original study, some lessons learned can be drawn about the PBR techniques. Because these studies were run in such different environments, we have more confidence in the robustness of the **external validity** of the conclusions (i.e. more confidence that they do not just hold in particular environments):

- Results from both experiments show that under certain conditions, the use of detailed procedural techniques can lead to improvements for subjects who have a certain minimum experience in the review perspective but are not yet experts in the inspection procedure. This was true both in the case of the NASA professionals with the relevant range of experience as well as of the Brazilian students. This provides more evidence to corroborate the theory that PBR is an effective way to bring novice reviewers up to speed, provided a certain minimal amount of previous training has been achieved.
- Both the original and the replication show benefits due to the detailed PBR techniques over less systematic approaches (the nonsystematic approach of NASA and the checklist approach at USP). This was true even when subjects had significantly more experience in the less systematic approach (as at NASA). This provides additional evidence that PBR can help improve inspection effectiveness for even experienced reviewers.

*Results*

The main results of this replication have been in the area of improving packaging. At the suggestion of the replicating research team, the original experimenters are trying to improve the packaging of the experiment by clarifying such items as the instructions for subjects, the time estimates for various tasks, and the descriptions of the known defects in the artifacts to be inspected.

At this point, thanks in part to a joint NSF/CNPq project, further replications have been run based on these experiences and the lessons learned. A total of four replications have now been conducted in Brazil, some at other universities (such as the Federal University of São Carlos) and others in industry. These have had the side-effect of highlighting another important benefit of replications: that they facilitate both the dissemination of the underlying concepts and technology and of the experimentation technology.

### Study at University of Maryland

The study described in this section is a consequence of a later study into the Perspective-Based Reading (PBR) approach to requirements inspection. Previous studies, such as those described in the earlier section on PBR, had shown that PBR was feasible and effective at detecting defects in certain environments. Our next goal had been to expand the understanding of the other variables that might affect the use of PBR. For that reason, a study was run in 1999 to collect observational data, at a very detailed level, about the use of the techniques.

One problem with that study was that conclusions about the technology were based on the experiences of "beginners," subjects who had only just learned about the technology and did not have significant experience in applying it. Thus one threat to validity was that we were measuring early effects that would increase or diminish over time, as subjects got more familiar with its use. Therefore we decided to undertake a replication **to address internal validity** that could explore such learning effects in more detail, and increase the confidence with which we could draw conclusions about the likely effectiveness of the process in industry.

*Subjects*

The subjects were graduate students at the University of Maryland. The subjects were paired up with one subject acting as the *executor* (responsible for applying the procedure) and the other as the *observer* (responsible for recording observations about the application). There were 26 subjects grouped into 13 pairs each subject got a chance to play each role so all 26 performed a review. As in the original study, around 1/3 of the subjects had industrial experience reviewing requirements.

*Procedure*

To study the technology at the level of detail that we were interested in, in both the original study and the replication, an observational approach was used. An observational approach is an research method suitable for understanding how a process is applied. In an observational study, a subject applies the technology being studied (the *executor* of the technology) while a researcher (the *observer*) observes them to understand details of the technology in use. Often the subject is instructed to "think aloud" so the researcher can better understand his or her thought processes. These types of studies provide a level of detail about individual process steps and their usefulness that is difficult to collect using traditional post-study questionnaires [Shull99]. The observational approach was necessary to understand what improvements might be necessary at the level of individual steps, for example whether subjects experience difficulties or misunderstandings while applying the technique (and how these problems may be corrected), whether each step of the technique contributes to achieving the overall goal, and whether the steps of the technique should be reordered to better correspond to subjects' own working styles.

Before the study, subjects received training in the reading techniques to be applied and in observational methods. In the first inspection, roughly half of the teams inspected the requirements for the LA and the other half the requirements for the PGCS. After this inspection was complete, the team members switched roles, i.e. the process observer in the first inspection became the process executor in the second inspection. The teams also switched requirements documents, from LA to PGCS or vice-versa. There was no team meeting to collect or detect more defects. The design is summarized in Figure 5.

| | **Original Study** | | **Replicated Study** | |
|---|---|---|---|---|
| **Activity** | *Teammate 1* | *Teammate 2* | *Teammate 1* | *Teammate 2* |
| **Training in PBR** | | | | |
| **Training in observational methods** | | | | |
| **Individual review using PBR** | (LA or PGCS) | (observed) | (LA or PGCS) | (observed) |
| **Individual review using PBR** | X | X | (observed) | (PGCS or LA) |

Figure 5: Order of activities in the study design. (Activities that did not occur in both the original and replicated studies are marked with an X and shaded in the appropriate place.)

## *Data Collection*

Quantitative data was collected, such as the time required to perform the inspection using PBR, and the number and type of defects detected. Using the observational techniques, a rich array of qualitative data was also collected, including:

- Subjective evaluation of the effectiveness of the technique.
- Specific problems with steps in the technique
- Usefulness of the different perspectives
- Practicality of the techniques (whether subjects would use some or all of them again)
- High-level problems with the techniques.

Because the team performed two requirements inspections, they were also able to include in their reports information about learning. Thus it was possible to evaluate whether the qualitative data was different because each team had an additional review during which to apply the techniques and would have learned from each other over the course of the study.

*Lessons Learned*

This study, when compared with the original, helped illuminate issues of **internal validity** by providing interesting results about our metrics for subject experience and learning.

- There may be other experience factors that affect the performance of inspectors when using the PBR techniques, because the results of the first and second studies showed opposite correlations in terms of experience vs. performance. In the replication, software development experience was negatively correlated to the performance in the inspection (unlike in the original study).
- There was no statistically significant difference in review effectiveness depending on whether the subject was reviewing a document from a familiar domain (PGCS) or an unfamiliar one (LA).
- There were no clear and consistent quantitative differences due to learning. For the each team's second review, both review effectiveness and effort required were about the same as the historical baseline for the document being inspected.

Although there were no quantitative indications of a learning effect, the qualitative data provided indications that the subjects were able to achieve the same review effectiveness in later reviews while short-cutting or otherwise modifying the techniques. The qualitative data provided the following results:

- 8/13 teams felt they improved as a result of the learning opportunity: they understood the steps better, were more confident and efficient the second time.
- 7/13 teams said that as a result of the learning opportunity they were able to change the application of the procedure to better match their own work practices, i.e. they could curtail or reorder steps when using PBR a second time.
- 6/13 teams provided new questions that could be asked during PBR (domain and organization specific)

*Results*

The main results of this study have been to propose a new set of hypothesis to be evaluated. Because the replication was the first one done in order to begin to understand learning, some questions could not be answered in the design of this study. Because we did not observe the improved performance in the second inspection that we hypothesized, we must investigate some of these other questions. The new questions include:

- Would we see a stronger learning effect if the same subject (as opposed to the same team) executed PBR two times?
- Is there some threshold of expertise that a subject must have in order to tailor effectively the PBR process for his needs?

### Lessons Learned about PBR

Both sets of replicated studies allow us to abstract up some common observations about PBR. Although not tested statistically, by combining data sets from all studies, we can formulate some general hypotheses based on patterns seen across multiple environments.

- Subject experience is not consistently related to a subject's effectiveness applying PBR. We have been trying to define heuristics that would allow developers to understand how much experience is necessary to apply PBR most effectively, but so far we have not been able to find consistent patterns that hold up across multiple studies, despite looking at several measures of experience. For example, in the section on the Study at The University of Maryland, despite the subjects having similar amounts of industrial experience, the relationship between software development experience and PBR effectiveness was opposite between the original and replicated studies. On the other hand, the study in the section on the Study at The University of Sao Paulo showed that earlier results with professional subjects were consistent with results seen from very much less experienced undergraduate students. Based on these results, we have observed a general trend that allows us to hypothesize that the PBR techniques may be best for novice training, and that the most experienced users get much less benefit if any from using PBR as compared to their usual approach. However, further testing is needed to refine this hypothesis and provide clear results.
- The structured review approach of PBR seems to help reviewers, even if the specific steps in the procedure aren't exactly right. In The Sao Paulo study, there were consistent results that systematic results were at least as good as and often better than less structured approaches. The observational study in the University of Maryland study provided one possible explanation why this could be true: a structured approach makes improvement easier because over time the steps can be updated based on the review teams' experience.

## *OORTs*

OORTs were first evaluated at University of Maryland [Travassos99] and have been the subject of a series of empirical studies with students and professionals through universities and industry in different countries [Shull01],[Melo01],and [Travassos02] that have demonstrated their effectiveness.

### Study at Norwegian University of Science and Technology

The replication [Arif01] was undertaken by an independent researcher in order to understand the feasibility of the techniques outside of the original environment, to see if the results of the original study [Shull99] were true only for the original set of subjects, or if the results were more globally applicable. In preparing to run the replication, the researcher hypothesized that the techniques were too detailed for easy use by subjects. The replicating researcher to vary some features of the techniques **for improvement**, removing certain details and moving the instructions to a more abstract level. The modifications to the techniques included:

- summarizing the detailed instructions of the original techniques into higher-level steps;
- adding more specific instructions to the reader to indicate the *type* of defect that was expected to be reported at different points in the techniques;
- changing the formatting.

The goal of these changes was to make the techniques more feasible to be performed in a shorter amount of time and produce more accurate and less ambiguous defect lists. Also, additional detail requested in the defect reports was to be used for checking the contribution of each step of the process to the final defect reports.

Researchers were novices in the technology (there was no one on site who had received training in the technology directly) at the Norwegian University of Science and Technology.

*Subjects*

There were 19 students who were members of a Master of Engineering class in Computer Science at the Norwegian University of Science and Technology, organized into 2- and 3-person teams. 10% had industrial experience in design reading.

*Procedure*

The replication made use of observational techniques, which are described in the University of Maryland Study, to examine the way in which subjects applied the less-detailed techniques. A modified version of the OO reading techniques from the original study was applied to the same artifacts used in the original study.

A design was used in which half of the class reviewed the LA design and the other half the PGCS, as shown in Figure 6. Five teams inspected the Parking Garage and 4 teams inspected the Loan Arranger design documents. The differences from the design of the original study were as follows:

1) In the replication, there was no requirements inspection prior to the design inspection, so there could be no analysis of whether familiarity with the system requirements had an impact on the effectiveness of the design inspection.
2) Subjects switched roles (executor and observer) in the middle of the study, rather than keeping the same role throughout.

Training in the observational techniques of the study was performed in the same way as in the earlier study.

Subjects received training in the techniques and observational study methodology, reusing the presentation slides of the original researchers. However, the class training and instruction were done by proxy, i.e. by an instructor who corresponded with the replicating researcher but had not participated in the training himself.

| | **Original Study** | | **Replicated Study** | |
|---|---|---|---|---|
| **Activity** | *Group A (half of subjects)* | *Group B (half of subjects)* | *Group A (half of subjects)* | *Group B (half of subjects)* |
| **Individual inspection of system requirements** | (LA or PGCS) | X | X | X |
| **Training in OORTs** | | | | |
| **Training in observational methods** | | | | |
| **Individual review using OORTs** | (LA or PGCS) | (LA or PGCS) | (LA) | (PGCS) |

Figure 6: Order of activities in the study design. (Activities that did not occur in both the original and replicated studies are marked with an X and shaded in the appropriate place.)

## *Data Collection*

Quantitative data was collected, namely the time required for executing the techniques and the number and type of defects detected. However, observational techniques were the most important method used in this study, providing data on:

- Executor's opinion of effectiveness of technique
- Problems encountered with specific steps of procedure
- How closely executors followed the techniques
- Practicality of the techniques (whether subjects would use some or all of them again)
- The problems encountered using the techniques

## *Lessons Learned*

Unfortunately, the quantitative data from this study did not allow us to assess the **improvement** due to the new version of the techniques. It was hoped that the replication would provide some indication of the effectiveness of the operationalized version of the techniques in comparison to the original version, but unfortunately that was not the case. The students did report a number of problems using the techniques, but it could not be determined if these were caused by changes introduced in the operationalized version, issues implicit in the original version and carried over, or problems that were uncovered while running the study. This problem actually led to many of the lessons learned that we formulated about running replications in general.

However, comparison of these results to the original study did allow us to confirm the effectiveness of some basic design features:

- Horizontal and vertical techniques found different types of defects (results were consistent with the original study).
- Having domain expertise was not helpful for subjects in the design inspection (results were consistent with the original study).

*Results*

Results from both studies are together contributing to a new version of the techniques, using the data from the observations about the way that readers applied the techniques. This version of the techniques will be focused more on the semantics behind the design models and less on the syntax. Additional improvements are being made in regards to training; lessons learned concerning the packaging necessary to support independent replications are discussed in a later section.

Study at the University of Southern California

At this point in their development, we had some confidence from multiple studies that the techniques were feasible and effective. The replication described in this section, in which the techniques were introduced into a course at the University of Southern California (USC), was one of our first attempts to have the techniques used on real development projects outside of the University of Maryland. It was especially important because several key differences existed between the two environments that required small focused changes to the techniques **for tailoring**.

These key changes from the original study [Travassos99] fell into the following areas:

- **Lifecycle Model***:* The students in the class used the Spiral lifecycle [Boehm88] model, whereas the original techniques were designed for use in a waterfall lifecycle. The Spiral model emphasizes iteration and incremental development, therefore the requirements and design are created concurrently and evolved over time.
- **Inspection Process***:* In these classes, the reading techniques were used in conjunction with a Fagan-style inspection process [Fagan86]. Previous studies had investigated the reading techniques in a different inspection context, in which the majority of defect detection was done during individual review. In the Fagan-style inspection process, the individual inspectors do some individual preparation, but use the team meeting as the primary time to detect the defects.
- **Inspected Artifacts***:* The students were also required to use a set of modeling and documentation guidelines called MBASE [Boehm99]. MBASE focuses on ensuring that a project's product models (e.g. architecture, requirements, code), process models (tasks, activities, milestones), property models (cost, schedule, performance, dependability), and success models (stakeholder win-win, IKIWISI—I'll Know It When I See It, business case) are consistent and mutually enforcing. Because the format of the artifacts created under MBASE was different from the format of the artifacts used in previous studies, some minor modifications had to be made to the reading techniques, to check conformance to the guidelines and cross-references to other MBASE documents.

Our hypothesis was that these changes could be made without destroying the feasibility of the techniques in the new environment. To test this hypothesis, one of the early OORT feasibility studies from UMCP was selected and replicated in the new classroom environment at USC. Other factors were held constant as much as possible in order to have a high degree of comparability between the studies.

The replication was done as a partnership between the developers of the OORTs and local researchers at USC, with both groups working together to adapt the techniques, the original researchers responsible for the classroom training and data analysis, and the local researchers responsible for data collection.

*Subjects*

The subjects were graduate students in the Computer Science Department at the University of Southern California enrolled in a two-semester Graduate Level Software Engineering Course. (The study described in this section took place in the Spring 2001 semester.) The majority of the 31 subjects had industrial experience (61% of the students had developed software as part of a team in industry).

*Procedure*

The replication differed from the original study in the assignment of subjects to designs. In the original study, all teams used the horizontal reading techniques to inspect their own designs to ensure that they were consistent. They corrected any defects that they found. After the designs had been corrected, the teams traded designs. Each team then performed the vertical reading techniques on a design for another team. The list of discrepancies found by the reviewers was then returned to the authors of the design for correction. In the study at USC, both horizontal and vertical reading techniques were applied by subjects to their own designs. In both of these inspections, team meetings followed individual review. These activities are summarized in Figure 7.

| Activity | Original study | Replication |
|---|---|---|
| **Team inspection of system requirements** | (own system) | (own system) |
| **Team inspection of design** | (own design, using horizontal techniques only) | (own system, using both horizontal and vertical techniques) |
| **Team inspection of design** | (another team's design, using vertical techniques only) | X |

Figure 7: Order of activities in the study design. (Activities that did not occur in both the original and replicated studies are marked with an X and shaded in the appropriate place.)

In the overall scope of the software development process there was no control group here. This occurred for two reasons: first, the design inspection was one small part of a larger study, and the overall design did not allow for a control group. Secondly, in a classroom environment, it was not possible to provide instruction on a topic to only a portion of the class.

The subjects used the Spiral development model to create their software. The OORTs were used in one development iteration to aid in the inspection of the designs.

The subjects used Fagan-style inspections in their projects. Unlike the other studies, the goal of the individual review was to prepare the individual reviewers for the main defect detection effort, which occurred at the team meeting. So, the individual inspectors used the OORTs in the preparation phase to help them make a list of potential defects, which they wanted to discuss during the team meeting. The team manager decided which subset of the techniques was relevant for the team and which techniques were assigned to which reviewers.

There was no control group. It was not possible, based on constraints of the class, to divide the class into a control and an experimental group. Also, pedagogically we could not teach the OORTs to only part of the class.

*Data Collection*

Questionnaires and an analysis of the defect lists were used to evaluate the effectiveness of the techniques in the development process. The quantitative data collected included both background information and the amount of time taken to use the techniques, used to evaluate feasibility of use. The qualitative data collected by the questionnaires concerned:

- Opinions of the helpfulness of the techniques.
- Problems encountered using the techniques, or extra knowledge that was needed to use the techniques.
- Opinions of effectiveness of training.

Analysis of the defect lists provided quantitative data about the number and types of defects found by the teams. The data was useful in determining if the output of the reading process uncovered defects and was useful for continuing the development process.

It was necessary to collect some additional information specific to the new context at USC, such as:

- Whether the techniques were able to be used with the specific design artifacts of the team;
- Whether any techniques were missing for a complete review of what the team felt to be important information.

*Lessons Learned*

The data from this study again showed us that the subjects were able to find real defects using the OORTs. Correction of these defects helped the subjects in their projects. The subjects also reported that they found the techniques useful and that the time required was not prohibitive in the context of the whole project. Most subjects thought the techniques were useful enough to recommend that they be used again.

More importantly, the data indicated that the techniques were **tailored** effectively to the new environment (Fagan-style inspections in a Spiral lifecycle model using the MBASE guidelines). This assessment was based on the results that:

- The researchers were able to analyze the environmental differences and make corresponding changes to the techniques;
- Subjects were able to use the techniques without reporting major difficulties and finding real defects;

- Qualitative feedback from the students reported that the training in the inspection techniques was one of the "5 most effective aspects" of the course.

As in the original study, the effort required for inspection using the techniques was not prohibitive. The average time spent preparing for the team meeting using the OORTs was 1.7 hours.

*Results*

Due to success of the replicated studies and the value that students in the class thought the training in the technology brought to them, researcherss from UMCP and USC are collaborating to train a local expert in the techniques at USC to continue the use and training of the techniques in education there. The local expert will be in a position to investigate further tailoring that might be needed of the techniques to the local environment. As a result, further studies will be undertaken to understand how the reading techniques can be integrated with MBASE and Spiral model to make for a more effective and unified development approach. One idea that we will concentrate on especially is whether the reading techniques can be used at different levels of detail at different points in the Spiral lifecycle, for example, to give a high-level review of the concepts at early iterations of the Spiral and at successively deeper levels of detail as the documents get fleshed out over time.

Lessons Learned about OORTs

As with PBR, both sets of replicated studies allow us to abstract up some common observations about the technology under study. Our current set of hypotheses about the OORTs, based on patterns seen across multiple environments, include:

- OORTs are feasible: All studies have confirmed that they do find real defects, that the different horizontal and vertical techniques are focused on different types of defects, and that the techniques can be used in a variety of environments.
- Some evidence has been provided to support hypotheses about the type of experience necessary for applying the techniques. For example, the surprising result from the original study that having a high level of experience in a domain was not beneficial during the inspection was confirmed in the replication. However, because of its counter-intuitive nature, we needed more evidence to support this hypothesis.
- Qualitative analysis of the difficulties seen across all studies show that the design methodology of the document author must be communicated to the inspectors to cut down on the number of defects raised because of different approaches to design. For instance, the type and level of detail of information that goes in the high-level design as opposed to a low-level design must be clearly understood. Design reviewers have to be

reminded of the possibility of multiple 'correct' designs for any given specification, and not to confused "defects" with "personal design preferences." At the same time, it should be recognized that some items reported by subjects, while not defects in the sense that they require a fix to be made, should be taken as advisory comments that are not necessary but could improve the quality of the design. Hence, even "false positives" reported can be valuable and should not just be discarded.

- Certain design choices and environmental assumptions may lead to process tailoring. The study at USC provided some evidence that such tailoring can be done effectively, at least when both the inspection techniques and the local environmental characteristics are sufficiently well understood.
- At this point in the evolution of the OORTs, we need controlled studies. A common thread in all the studies reported in this section has been the comparison of one version of the OORT techniques to a previous version, in order to test whether changes for improvement or tailoring have been done effectively. Based on this series of studies, the techniques have become quite sophisticated compared with the earliest versions and many of the original issues have been addressed. It is our subjective evaluation that this body of data shows that the techniques are ready to be compared to other forms of design inspection approaches for evaluation.

*Lessons Learned about Reading Techniques*

By combining the results of multiple replications discussed in this chapter, we were able to abstract up some common observations about reading techniques in general that are supported by some evidence, across environments, studies, and in fact specific families of reading techniques being studied. Although at a high level of generality, lessons learned at this level have helped us understand the common basis of effective reading techniques and further tailor that basic approach to new types of inspections and new environments.

- A procedural approach to individual review is a feasible way both to find defects and to focus reviewers' attention on different aspects of the document under review. In all studies it was shown that:
    - All reviewers reported issues that represented real defects (not just differences of perspective or opinion);
    - Reviewers found different types of defects depending on the specific reading techniques in the family that they were applying
- Experience measures are hard to match to process recommendations. The lack of a consistent correlation between any experience measure and review effectiveness is itself a result of these studies. It remains to further study to show whether this is because there simply is no correlation between effectiveness and experience, or because we haven't yet found an appropriate way to measure the relevant experience. We are continuing to

explore several hypotheses (e.g. experts don't benefit as much from introducing reading techniques, while novices with sufficient background benefit the most). Such hypotheses have definite implications for practice if we can build an acceptable level of confidence: For example, they could show that reading techniques can help with cross training (i.e. bringing experts up to speed on each other's roles), or can help novices fill in when the experts aren't available to participate in reviews.

- We have varying degrees of confidence about several different design principles for developing reading techniques. Going back to the three "best practices" listed earlier, we have collected some information validating that these are good design principles:
    - Focused perspectives: Results across all studies described in this chapter are clear that focused perspectives are feasible and effective. When compared to one another, it was shown that different techniques found different things (showing that focusing of a reviewer's attention can be done effectively). When compared to checklist or ad hoc review approaches, it was shown that the focused techniques did better than unfocused inspection approaches.
    - Active review: The benefits of an active review where subjects have to work with and somehow manipulate the information in the document have never been tested separately in a controlled study. However, some qualitative evidence has been collected to address the issue: some participants have reported feeling more comfortable in following directions to work on intermediate models of the information rather than being given no guidance as to exactly how to check the information in the document. Studies are needed that can adequately test this hypothesis, e.g. by looking at how do people perform when not working with models during review.
    - Defect taxonomies: Although it is intuitive that making defect taxonomies explicit helps improve inspections by telling reviewers more clearly what to look for, there has been no direct indication of the effectiveness of this approach from the replicated studies described here. This approach has always confounded with other design principles, because in the context of the studies it was not a fair comparison to ask people to use different approaches but give them different amounts of guidance as to what to search for.

## Lessons Learned about replication

Based on the four replications discussed in the previous section, in addition to the lessons we learned about reading techniques we were also able to learn some lessons about running replications. We report those lessons here as a guide for researchers running replications in the future.

- Having a local expert on hand who is well versed in the technology being studied is crucial.
- Pilot studies by the replication researchers can be important for understanding and conveying information about the study and proper expectations to the subjects (e.g. concerning time requirements for participation).
- When replicating a study, improving the training by allowing the subjects a laboratory setting in which to practice is seen as positive by the subjects.
- "Modular" designs, such as the one used to support the replication in Sao Paulo, can facilitate replications by including both generic and environment-specific parts. With this type of design, it was easy to adapt the experiment for use in a non-NASA environment since environment-specific treatments could be removed from the design without diminishing the overall ability to investigate the hypotheses.
- When the subject of the technology is "requirements defects" it can be hard to communicate clearly enough what the issues are with the requirements document, in order to enable the replicating researchers to identify them specifically in the subjects' defect reports. Subjects have many, many ways of expressing the items they find on their defect reports, and deciding whether or not the reported items represent real defects can be difficult.
- It is hard to understand how different subjects may interpret the same instructions: It became clear in retrospect that the Brazilian students did not report some defects that they found but we would have liked to have studied, because they noticed those defects when not applying a step of the inspection procedure and thought we only cared about defects directly related to the procedure. There were also terminology problems: The word "scenario" was confusing in Brazil because they had previously learned a specific definition of the term for use case creation, and it was applied less formally in another context during the training. Another example was the word "service," which was used in the OORTs to represent one level of design abstraction but has a specific meaning in the telecom industry. Terminology must be clearly defined and explained to subjects.

The last point demonstrates how hard it is to do an exact replication of a study, even when the original and replication research teams are working closely together. There are many misunderstandings or simply different interpretations that can occur on the part of subjects due to cultural or domain issues, and trying to identify them all beforehand is almost impossible.

Because of the above lessons learned, we have formulated the following guidelines that we feel are crucial for getting the planned type of results from a replication:

- *Choosing the right type of replication is necessary for running an effective study.* A first step in preparing for any replication should be to analyze the replicating environment in order to understand if tailoring of the technology is necessary. At the point in the development of the technology at which this replication was run, it is not clear what aspects were important and what aspects could be altered for improvement. Therefore, if the changes are driven by specific environmental factors, the replication will have a better chance of success.
- *Don't minimize the importance of "training the trainer."* Effective training of the researcher who is going to perform the replication is crucial. When there is no local expert and the expectations are not clearly communicated, problems arise that could most likely be avoided.
- *An effective replication requires packaging the object of study with key context information, not just packaging the study design itself.* It is clear from this study that packaging a study for replication requires a lot of work on the part of the original researchers. The study package requires much more than just the artifacts used in the study. It is difficult to capture all of the knowledge needed to replicate a study.

## *Moving toward greater formality in building bodies of knowledge*

We stated earlier that the field is not yet sophisticated enough for us to build bodies of knowledge using formal statistical methods. Based on our experiences so far, however, we can begin to formulate a potential meta-data description of the studies that would provide a necessary step towards this goal. This meta-data would describe the context in which a study has been run, capturing the important aspects of different replications on the same topic. Such meta-data would allow readers to understand the relevance of studies to their own environment and what contributions they make to the larger body of knowledge. Both the study data and the meta-data must be used together to draw conclusions about the technology.

Capturing meta-data in a comparable way across multiple environments requires, first, that the environments can be meaningfully compared and second, that the same definitions of the meta-data variables are used for each environment. Describing the environments in a comparable way requires measuring at a comparable level of granularity, i.e. measuring equivalent organizational units. For example, there is probably no way to describe "the" software development process at a large organization that incorporates a multitude of different development projects, each with its own technologies used and set of quality concerns. Rather, the meta-data should be measured at the level of individual projects. That is, each set of meta-data should describe one project, not larger heterogeneous organizational entities like divisions or departments.

The types of questions that could be answered in a meta-analysis of software technology data basically concern three sources of context variation:

- Of the software development technology. (Is a particular change to the development process more or less effective than some alternative process?)
- Of the systems to which the technology is being applied. (Are different types of artifacts, notations, or systems more or less suited to different development technologies?)
- Of the personnel applying the technology to systems. (Do subjects with different levels of experience or skills apply the technology differently?)

Choosing meta-data that describe each of these aspects of an environment limits the specific questions that can be answered by the analysis, so it is important to choose the meta-data wisely. Choosing a common set of meta-data should take into account well-supported ideas for what variables are responsible for affecting software development effectiveness.

Describing a technology is difficult, but can often be addressed by describing as specifically as possible what version of a given tool, process, or methodology was used.

To describe the application profile of the systems to which the technology is applied, we have identified the following set of initial meta-data:

- Application Context: The description should contain information on whether the object of study was used in an <u>isolated (or classroom) development task</u> not done as part of the development of a real project (e.g. a requirements inspection of pre-existing documents, with no action taken on the resulting defect lists) or as part of an actual <u>development project</u>.
- Size of project: Maximum team size required to complete the project.
- Type of application: Brief description of the domain of the application. May include a link to a prepackaged application description.
- Platform: Values might include: Mainframe, client/server, network, applications generator. (This list should be extended as needed.)
- Process Drivers (ranked): What were the primary quality factors of concern to the subjects? To answer, the quality factors from the following set should be listed in order as appropriate: Requirements, cost, schedule, dependability, option exploration.

- Time Period: Include the start and end dates, and any key milestones.

The subjects applying the object of study can be described by means of the following attributes:

- Developer skill and experience: Were the subjects students or professionals? The number of years of experience can also be measured in certain, technology-specific areas. For example, in general we could measure a subject's number of years of industrial experience (although because of wide interpersonal variation this is usually not so useful). For requirements inspection technologies, we might also measure the number of years (or number of projects) experience with inspections, with requirements inspections specifically, and with creating requirements.
- Relative skill level: Subjective assessment of how subjects compare to other subjects in similar types of populations.

Example Instantiations

As a practical way to avoid single isolated studies, and to allow these expensive undertakings to contribute to the larger body of knowledge, the United States' National Science Foundation funded the Center for Empirically Based Software Engineering (CeBASE) in 2000. CeBASE has as part of its mandate the role of improving software development by communicating to developers what heuristics and models exist to support decision-making based on empirical research. CeBASE has researched methods for abstracting and modeling the needed information for decision support across multiple studies, and collaborates on further empirical studies where necessary to support that research. We are refining our conception of packaging by creating a repository of studies to facilitate replications in the area of reading techniques. This repository, currently still under construction, is available to CeBASE affiliates. See http://www.cebase.org/www/researchActivities/defectReduction/index.htm for more information. A related repository based on a similar setup is also being created for the Simula Research Lab in Norway, http://www.ifi.uio.no/isu/forskerbasen. The ViSEK project has a similar mandate to create an experience base of empirically validated technologies for German industry (www.visek.de). Finally, the Experimental Software Engineering Research Network (ESERNET, www.esernet.org) is also interested in families of related studies that can be used to abstract higher-level conclusions.

## Conclusions

In this chapter we have discussed a family of reading techniques useful for the individual inspection (reading) of software documents. Specifically, we have focused the discussion around a technique for inspection of requirements (PBR) and a technique for inspection of object-oriented design (OORTs). A series of empirical studies have been run on these two sets of techniques in order to evolve and improve them.

This chapter has provided a discussion of the latest series of studies that we have run on PBR and OORTs. Based on these studies, one of the key conclusions is that there is a need to analyze the results of these studies collectively. Isolated studies only provide part of the picture, but families of coordinated studies allow researchers to build up bodies of knowledge about development technologies. In this chapter we have provided a brief discussion and justification for the need to build bodies of knowledge.

One of the main techniques that we have found useful in building these bodies of knowledge is that of replications. We have explained how replications can be useful in building bodies of knowledge. In conjunction with that we provided a discussion of the various types of replications and when each type should be used.

These different types of replications are then illustrated by a discussion of the studies used to evolve and improve PBR and OORTs. For each of the replications, we have provided some justification as to why the particular type of replication was chosen. At the end of each discussion, we have discussed the results learned from the replication that could not have been learned from a single study. We conclude the discussion of the replications by providing lessons learned about PBR, OORTs and reading techniques in general from the combination of all the replications that we could not have learned from any single replication.

We then have concluded the chapter by discussion how the process of replications can be improved. Our experiences have provided us with several guidelines concerning how and how not to run replications. The main vehicle for successful replications is the lab package. We have shown the necessity of such packages, and also given some discussion as to how researches begin to build these lab packages.

## References

[ANSI84] ANSI/IEEE. “IEEE Guide to Software Requirements Specifications.” Standard Std 830-1984, 1984.

[Arif01] Arif, T., and Hegde, L.C. “Inspection of Object Oriented Construction: A study of Reading Techniques tailored for inpection of Design Models expressed in UML.” Prediploma Thesis, Norwegian University of Science and Technology, Nov. 2001. Available at http://www.idi.ntnu.no/grupper/su/sif8094-reports/p2-public.pdf

[Basili96] Basili, V.R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sorumgard, S., and Zelkowitz, M.V. “The Empirical Investigation of Perspective Based Reading.” *Empirical Software Engineering—An International Journal*, 1(2): 133-164, 1996.

[Basili97] Basili, V. “Evolving and Packaging Reading Technologies.” *The Journal of Systems and Software*, 38(1): 3-12, July 1997.

[Basili99] Basili, V., Shull, F., and Lanubile, F. “Building Knowledge through Families of Experiments.” *IEEE Transactions on Software Engineering*, 25(4): 456-473, July 1999.

[Boehm88] B. Boehm. “A Spiral Model of Software Development and Enhancement.” *IEEE Computer* 21(5): 61-72, May 1988.

[Boehm99] Boehm, B., Port. D., Abi-Antoun, M., and Egyed, A. “Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) deliverables for Model-Based Architecting and Software Engineering (MBASE).” USC Technical Report USC-CSE-98-519, University of Southern California, Los Angeles, CA, 90089, February 1999.

[Ciolkowski97] Ciolkowski, C., Differding, C., Laitenberger, O., and Muench, J., “Empirical Investigation of Perspective-based Reading: A Replicated Experiment.” International Software Engineering Research Network, Technical Report ISERN-97-13, 1997. http://www.iese.fhg.de/ISERN/technical_reports/isern-97-13.pdf

[Fagan76] Fagan, M. E. "Design and Code Inspections to Reduce Errors in Program Development." IBM Systems Journal, 15(3):182-211. 1976.

[Fagan86] Fagan, M. E.. "Advances in Software Inspections." *IEEE Transactions on Software Engineering*, 12(7): 744-751, July 1986.

[Gilb93] Gilb, T. and Graham, D. *Software Inspection*. Addison-Wesley, Reading, MA, 1993.

[Knight93] Knight, J.C. and Myers, E.A. "An Improved Inspection Technique." *Communications of the ACM*. 36(11): 51-61, Nov. 1993.

[Lewis95] Lewis, T., Rosenstein, L., Pree, W., Weinand, A., Gamma, E., Calder, P., Andert, G., Vlissides, J. and Schmucker, K. *Object-Oriented Application Frameworks*. Mannings Publication Co., Greenwich, 1995.

[Melo01] Melo, W., Shull, F., and Travassos, G.H. "Software Review Guidelines." *Systems Engineering and Computer Science Program -PESC ES-556/01*. COPPE/UFRJ. September, 2001. http://www.cos.ufrj.br/publicacoes/reltec/es55601.pdf

[Miller00] Miller, J. "Applying Meta-Analytical Procedures to Software Engineering Experiments." *Journal of Systems and Software*, 54(1): 29-39, 2000.

[Porter95] Porter, A., Votta Jr., L., and Basili, V. "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment." *IEEE Transactions on Software Engineering*, 21(6): 563-575, June 1995.

[SEL92] Software Engineering Laboratory Series. *Recommended Approach to Software Development, Revision 3*. SEL-81-305, pp.41-62, 1992.

[Shull98] Shull, F.S. "Developing Techniques for Using Software Documents: A Series of Empirical Studies." *Ph.D. Thesis,* Computer Science Department, University of Maryland, 1998.

[Shull99] Shull, F., Travassos, G.H., Carver, J., and Basili, V. "Evolving a Set of Techniques for OO Inspections." Technical Report CS-TR-4070, UMIACS-TR-99-63, University of Maryland, October 1999. http://www.cs.umd.edu/Dienst/UI/2.0/Describe/ncstrl.umcp/CS-TR-4070

[Shull01] Shull, F., Carver, J., and Travassos, G.H. "An Empirical Methodology for Introducing Software Processes." In *Proceedings of European Software Engineering Conference*, Vienna, Austria, Sept. 10-14,2001. pp. 288-296.

[Shull02a] Shull, F. "Software Reading Techniques." In the Encyclopedia of Software Engineering, Second Edition. Copyright John Wiley & Sons. 2002.

[Shull02b] Shull, F., Basili, V., Carver, J., Maldonado, J., Travassos, G. H., Mendonça, M., Fabbri, S. "Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem." Accepted at the International Symposium on Empirical Software Engineering 2002, Nara, Japan, October 2002.

[Sørumgård97] Sørumgård, S., *Verification of Process Conformance in Empirical Studies of Software Development*, Ph.D. Thesis, Norwegian University of Science and Technology, February 1997, Chapters 10-11. http://www.idt.unit.no/~sivert/ps/Thesis.ps.

[Travassos99] Travassos, G.H., Shull, F., Fredericks, M., and Basili, V.R. "Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality." *OOPSLA'99*. Denver, CO, Nov. 1999.

[Travassos02] Travassos, G.H., Shull, F., Carver, J., and Basili, V.R. "Reading Techniques for OO Design Inspections." Technical Report CS-TR-4353, UMIACS-TR-2002-33, University of Maryland, 2002, 56 p. URL: http://www.cs.umd.edu/Library/TRs/. Also available at http://www.cos.ufrj.br/publicacoes/reltec/es57502.pdf

[Votta93] Votta Jr., L.G. "Does Every Inspection Need a Meeting?" *ACM SIGSOFT Software Engineering Notes*, 18(5): 107-114, December 1993.

[Wood99] Wood, M., Daly, J., Miller, J., Roper, M. "Multi-method research: An empirical investigation of object-oriented technology." *Journal of Systems and Software* 48(1): 13-26, 1999.

[Zhang99] Zhang, Z., Basili, V., and Shneiderman, B. "Perspective-based Usability Inspection: An Empirical Validation of Efficacy." *Empirical Software Engineering: An International Journal* 4(1): 43-70, March 1999.