Empirical Software Engineering, 9, 111–137, 2004.

Knowledge-Sharing Issues in Experimental Software Engineering

FORREST SHULL fshull@fc-md.umd.edu
Fraunhofer Center for Experimental Software Engineering, 4321 Hartwick Road, Suite 500, College Park, MD 20740

MANOEL G. MENDONÇA mgmn@unifacs.br
Computer Networks Research Group (NUPERC), Salvador University (UNIFACS), Rua Ponciano de Oliveira, 126 Salvador, BA 41950-275, Brazil

VICTOR BASILI basili@fc-md.umd.edu
Fraunhofer Center for Experimental Software Engineering, Department of Computer Science, A.V. Williams Building, University of Maryland, College Park, MD 20742

JEFFREY CARVER carver@cs.umd.edu
Department of Computer Science, A.V. Williams Building, University of Maryland, College Park, MD 20742

JOSÉ C. MALDONADO jcmaldon@icmc.sc.usp.br
Instituto de Ciências Matemáticas e de Computação, Departmento de Ciências da Computação e Estatística, Av. Trabalhador São-Carlense, 400 – Centro Caixa Postal 668, 13560-970 São Carlos, SP, Brazil

SANDRA FABBRI sfabbri@dc.ufscar.br
Universidade Federal de São Carlos, Departamento de Computação, Rodovia Washington Luiz, km235, Caixa Postal 676, 13565-905 São Carlos, SP, Brazil

GUILHERME HORTA TRAVASSOS ght@cos.ufrj.br
Rua Primeiros Sonhos 101/304, Jardim Guanabara – Ilha do Governador, Rio de Janeiro, RJ, Brasil, 21941-240

MARIA CRISTINA FERREIRA cristina@icmc.usp.br
Instituto de Ciências Matemáticas e de Computação, Departamento de Ciências da Computação e Estatística, Av. Trabalhador São-Carlense, 400 – Centro, Caixa Postal 668, 13560–970 São Carlos SP, Brazil

Editor: Ross Jeffery

Abstract. Recently the awareness of the importance of replicating studies has been growing in the empirical software engineering community. The results of any one study cannot simply be extrapolated to all environments because there are many uncontrollable sources of variation between different environments.

In our work, we have reasoned that the availability of laboratory packages for experiments can encourage better replications and complementary studies. However, even with effectively specified laboratory packages, transfer of experimental know-how can still be difficult. In this paper, we discuss the collaboration structures we have been using in the Readers' Project, a bilateral project supported by the Brazilian and American national science agencies that is investigating replications and transfer of experimental know-how issues. In particular, we discuss how these structures map to the Nonaka–Takeuchi knowledge sharing model, a well-known paradigm used in the knowledge management

literature. We describe an instantiation of the Nonaka–Takeuchi Model for software engineering experimentation, establishing a framework for discussing knowledge sharing issues related to experimental software engineering. We use two replications to illustrate some of the knowledge sharing issues we have faced and discuss the mechanisms we are using to tackle those issues in Readers' Project.

Keywords: Empirical software engineering, experimental design, experimental replication, software reading techniques.

1. Introduction

In the past few years, there has been a growing awareness in the empirical software engineering community of the importance of replicating studies (e.g. Brooks et al., 1996; Johnson, 1996; Lott and Rombach, 1996; Basili et al., 1999; Miller, 2000; Shull et al., 2001). Most researchers accept that no one study on a technology should be considered definitive. Too many uncontrollable sources of variation exist from one environment to another for the results of any study, no matter how well run, to be extrapolated to all possible software development environments. The goal is to build a consolidated, empirically based body of knowledge that identifies the benefits and costs of various techniques and tools to support the engineering of software.

A result of this realization is an increased commitment to run more studies in a variety of environments. Replication in different environments is an important characteristic of any laboratory science. It is the basis for credibility and learning. Complementary, replicated studies allow researchers to combine knowledge directly or via some form of meta-analysis. Since intervening factors and threats to validity can almost never be completely ruled out of a study, complementary studies also allow more robust conclusions to be drawn when related studies can address one another's weak points. In software engineering, this replication process enables us to build a body of knowledge about families of related techniques and basic principles of software development.

It is important to note that for the purposes of building up a body of knowledge, ''replication'' needs to be defined relatively broadly: 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 we will consider a replication to be a study that is run, based on the design and results of a 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 result of a study in a different context, then the design of the original study may be slightly modified but still considered a replication.

In our own work, we have reasoned that better replications and complementary studies can be encouraged by the availability of laboratory packages that document an experiment. A *laboratory package* describes an experiment in specific terms and provides materials for replication, highlights opportunities for variation, and builds a context for combining results of different types of experimental treatments. Laboratory packages build an experimental infrastructure for supporting future replications. They establish a basis for confirming or denying original results,

complementing the original experiment, and tailoring the object of study to a specific experimental context.

However, despite our high hopes, our experience has shown that replication is difficult and lab packages are not the solution by themselves. Even when both the original researchers and the replicating researchers are experienced experimentalists, there are so many sources of variation and implicit assumptions about the experimental context that composing a static lab package to describe all relevant aspects of the experiment, in such a way that unexpected sources of variation are not introduced into the replication, is nearly impossible. As examples, consider the following cases from our own experience:

- A study of a software review technique unintentionally introduced a source of variation when a limit was placed on the time available for performing the review (in the original experiment, the time was open-ended). The results were very different (and incomparable) between the two studies because subjects in the replication altered their behavior to try to prioritize aspects of the review and could not check their work, while subjects in the original study were allowed to work under more realistic conditions.

- Another study of software review techniques introduced a possible variation in results when the wrong time estimate was given to reviewers. When the review took much longer than the reviewers had expected, they reported feeling frustrated and de-motivated with the technique and quite possibly reviewed less effectively as a result.

The point of these examples is not to imply that either the original or replicating researchers were somehow deficient in their preparation,[1] but to illustrate the major changes to experimental results that can occur from seemingly miniscule changes to experimental conditions. Thus the issue of process conformance between seemingly identical experiments becomes a real issue that may affect the level of confidence in results.

In a previous paper (Shull et al., 2002), we have reviewed our experiences in experimental replication leading us to identify an important tacit knowledge problem affecting the process conformance and hence the comparability of results between replications. (Here, tacit knowledge refers to information that is important to the experiment but is not coded into the lab package. It lies in the heads of the original experimenters and is difficult to make explicit, for a variety of reasons.) As argued in that paper, we believe that effective replications require not only lab packages but also a process of replication involving the original researchers to support package instantiation, evolution, and use. The need for such a process implies an associated need for an effective collaboration structure between the original and replicating researchers to convey both the explicit (lab package) and the tacit knowledge that is important for the experimental process conformance.

In this paper, we discuss the collaboration structures we have been using in the Readers' Project, a bilateral project supported by the Brazilian and American

national science agencies that is investigating replications and tacit knowledge issues. In particular, we discuss how these structures map to the Nonaka–Takeuchi knowledge sharing model, a well-known paradigm used in the knowledge management literature (Nonaka and Takeuchi, 1995). We instantiate the Nonaka–Takeuchi Model for software engineering experimentation, establishing a framework for discussing knowledge sharing issues related to experimental software engineering. We use two replications to illustrate some of the knowledge sharing issues we have faced and discuss the mechanisms we are using to address those issues in the Readers' Project.

The rest of this article is organized as follows. Section 2 provides a brief introduction to the Readers' Project. Section 3 describes Nonaka and Takeuchi's knowledge sharing model and instantiates it for experimental software engineering. Section 4 discusses two replications of an experiment, paying special attention to the knowledge sharing issues in these replications. Section 5 discusses the mechanisms we have been using in the Readers' project to facilitate knowledge sharing, learning, and knowledge evolution. Finally, Section 6 summarizes the points raised and articulates a set of issues that should be addressed for experimentation improvement.

2. Introduction to the Readers' Project and Research Questions in Knowledge Sharing

The high level goal of the Readers' Project is to build an empirically based body of knowledge about software analysis techniques in different cultures by facilitating the running of more and better replications. These replications will allow us to support more useful and robust conclusions concerning the effectiveness of these techniques.

To address issues of running effective replications, a cooperation between Brazilian and American researchers, named "Readers: A Collaborative Research to Develop, Validate and Package Reading Techniques for Software Defect Detection," was initiated in 1999. This cooperation, supported by the Brazilian (CNPq) and American (NSF) national science agencies, has the general purpose of investigating techniques for analyzing software documents in diverse cultural settings. The approach taken in this research is to tailor these software analysis techniques to accomplish a specific software-related task (defect detection) using specific kinds of software documents (requirements, specification, and code documents) in specific environments or cultural settings (Brazilian and American industries and academia) based on empirical evaluation. The members of the project contribute crucial experience in the areas of both developing and refining software technologies, and in empirically evaluating technology effectiveness.

Running more studies does not simply require more researchers interested in empirical studies. Empirical studies are complex and very difficult to run in isolation without a community of like-minded researchers to give feedback and suggest new directions. There is a need for:

- Mechanisms for creating a community of researchers (e.g. the Center for Empirically Based Software Engineering—CeBASE (Goth, 2001)).

- Fostering of specific collaborations (e.g. replicating experiments in a specific domain).

- Transfer of experimentation know-how (e.g. producing reusable laboratory manuals).

In the Readers' Project a long-range goal is to contribute to the advancement of the empirical community, particularly by addressing the last two points.

Since there is no "perfect" study that removes every conceivable threat to validity, we must learn new techniques for combining studies so that more robust results may be achieved across several empirical sources. Moreover, different types of studies are not interchangeable; more sophisticated techniques are needed and their relevance for different types of situations must be better understood. For these reasons, the Readers' project is concerned with more than just running replications. The Readers' project aims to create and explore:

- *Techniques for reaching higher-level conclusions about a technology*. Drawing "better" conclusions in the sense that they are more robust, justifiably inspire more confidence, or address useful levels of generality requires conclusions to be drawn across families of studies, rather than from individual studies (Basili et al., 1999). One area of research we will explore is how to compose an effective family of studies: that is, given the weaknesses of existing studies, which new studies would be most optimal? Initial work in this area has produced procedures for developing well-grounded hypotheses from multiple data sets, by comparing each new set of data to the existing state of the knowledge to iteratively refine hypotheses (Carver, 2003).

- *Guidelines for packaging the experimental artifacts*. Laboratory packages have been found to be an important tool for supporting replications (Shull et al., 2002). Not only do lab packages support more replications (easily-available designs and materials facilitate replications by reducing the amount of effort required from independent researchers) but well-designed packages are crucial for facilitating better, comparable replications. We have already identified some items that are needed in an effective lab package and missing in existing lab packages (Shull et al., 2002; Amaral and Travassos, 2003).

- *Procedures and tools for evolving lab packages*. A major difficulty in running replications is artifact evolution. Given the large numbers of documents that comprise any lab package and the interrelationships between those documents, maintaining consistency and comparability of results between experiments over time imposes significant overhead effort. (For example, if an inspection experiment uses a requirements document with seeded defects, that document can evolve but always has to be matched up with the correct list of seeded defects, the correct version of the system design, and the appropriate results that were calculated based on a particular version of the defect list.) Finding ways to

maintain these complex relationships between different versions of documents will help decrease the effort required of the researcher and increase the level of confidence in results. The evolution of the lab packages in the Readers' project is a constant, ongoing activity.

These techniques, guidelines and procedures will help us better understand and define the knowledge transfer process. But, effective replications require more research into the knowledge transfer process itself. For example, what kinds of collaboration mechanisms are necessary and sufficient for the information about a single experiment to be transferred from one researcher to another? What are the main steps in running an experiment and what information is necessary at each step?

When the relevant information is not correctly transferred, misconceptions caused by bad assumptions about experiment design can have far-reaching impacts on the success of the replication (Shull et al., 2002). Focused working groups, physically co-located with all relevant experimental materials on-hand, seem best suited to combating these misconceptions. Meetings support discovery of the many inter-related facets of an experiment in a way that shorter, focused communication exchanges do not. But, it is not always feasible to have a close collaboration, so defining the experimentation process and assessing process conformance during the replication become mandatory activities to ensure successful results. Defining the experimentation process aims at making clearer and explicit the tacit knowledge—the knowledge and assumptions underlying the techniques and the related Laboratory Package—as discussed in the next section.

3. Knowledge Sharing Issues in Experimental Software Engineering

Even when effectively specified packages exist, researchers face difficulty understanding and reviewing the available laboratory packages to select an appropriate one. The main difficulty (found also in the replications described later in this paper) is to understand the concepts underlying the techniques under study and to master the knowledge involved in running the experiment. This problem can be thought of as the difficulty in transferring experimental know-how between the original experimenters—the knowledge providers—and the replicators—the knowledge users. Two kinds of knowledge must be transferred in this process: tacit knowledge and explicit knowledge. The explicit knowledge refers to information that is written down within the laboratory package. The replicators can absorb this kind of knowledge by reading and using the lab package. The tacit knowledge refers to information that is important to the experiment but is not coded into the lab package. This tacit knowledge is often crucial for the interpretation of the explicit knowledge.

3.1. Knowledge Sharing in Related Literature

Empirical software engineering is not unique in facing the problem that important information can still be tacit and hence difficult to convey to other members of the community. Tacit knowledge problems have been identified in similar fields that set out to solve complicated technical problems. Nonaka and Takeuchi (1995) proposed a knowledge-sharing model—also called the ''tacit-explicit model''—in which the knowledge-sharing process is illustrated (Nonaka and Takeuchi, 1995). As seen in Figure 1, the model portrays the knowledge-sharing problem as a continuous cycle. It starts with socialization, in which tacit knowledge is transferred between individuals. It progresses to externalization, in which individuals make their tacit knowledge explicit to the group or organization. Through combination, explicit knowledge is then organized into more sophisticated or abstract knowledge. And lastly, through internalization individuals will absorb explicit knowledge, combining it with their own knowledge and experiences to produce new tacit knowledge. The cycle fosters continuous learning and knowledge gain.

Another way to look at the problem is to consider the knowledge life cycle (Wiig, 1999; Agresti, 2000; Dixon, 2000). Knowledge evolves through a path that goes from knowledge creation or acquisition (in which tacit knowledge is made explicit), to its organization and storage, to its distribution to potential users, to its application by those users. The application of knowledge can become the basis for new knowledge creation, beginning a new cycle (Lindvall et al., 2002).

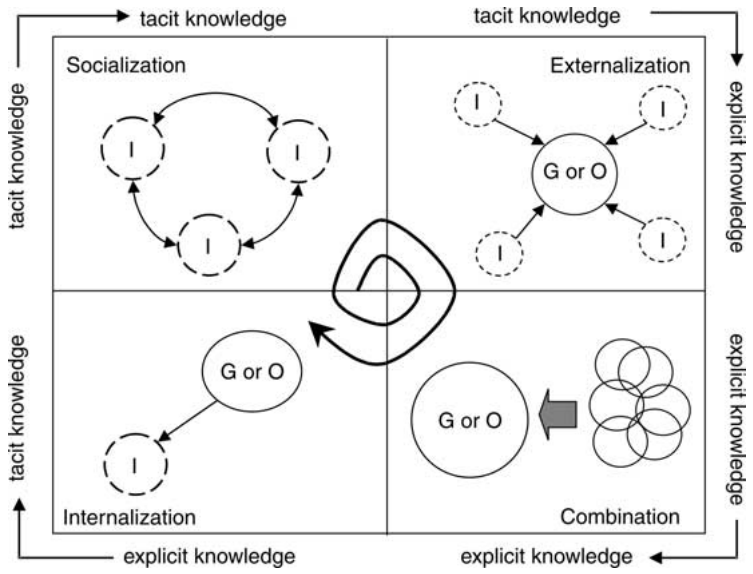

Figure 1. Knowledge-sharing model, adapted from Nonaka and Takeuchi (1995). "I" represents an individual; "G or O" represents a group or organization.

Knowledge sharing issues have also been identified in connection with software engineering. Rus et al. (2001) have produced a survey on the subject (Rus et al., 2001) that argues that developing software is a "design type process" where every one of the (possibly many) people involved has a large number of decisions to make. In order to develop complex software systems, then, these individuals need to communicate and coordinate, so that individual knowledge can be "shared and leveraged at a project and organization level."

3.2. Knowledge Sharing in Empirical Studies

In the field of empirical software engineering, a central challenge is to efficiently share experimental knowledge between replicators in order to facilitate replication comparability. Considering Figure 1, the problem can be split into four phases: socializing knowledge among replicators, making tacit knowledge explicit, improving explicit knowledge, and internalizing experimental knowledge.

Running a replication at one site, over and over again, facilitates socialization and internalization of experimental knowledge because the sharing of tacit knowledge is easier. Running replications over a larger community requires addressing the tacit knowledge problem better.

While the goal of the original experimenter is to make as much tacit knowledge explicit as possible, it is important to observe that some tacit knowledge is not made explicit because the experimenter does not anticipate its relevance to other researchers, or simply because it seems so minor that he or she did not think it was an important enough component of the overall experimental design to describe. One example is the decision process involved in updating some aspect of the lab package. For example, is there a process for updating the list of defects related to a specific artifact used in an experiment? What do you do when a new defect has been identified? When should the list of defects be updated, and how? Who should be involved? However, for other kinds of tacit knowledge it is too difficult or impossible to make it explicit. This knowledge includes tips, simple procedures and decisions that are too detailed to write into a lab package. For capturing this kind of information, it is very important for the replicators to interact with researchers who have run the experiment before.

Absorbing tacit knowledge has been a major source of difficulty during the replications done so far in the Readers' Project. The lack of communication of tacit knowledge can affect the comparability of a replication's results to the original experiment. When different and independent researchers carry out replications, there is a danger that experimental steps (such as training) are executed in slightly different ways that introduce new sources of variability. Systematic procedures must be stated and followed for the creation and use of lab packages. These procedures must establish, in a step-wise fashion, each task that must be accomplished to run the experiment, including timing and deliverables. Still, it is our opinion that interaction between the replicators and the original experimenters is fundamental to ensure the

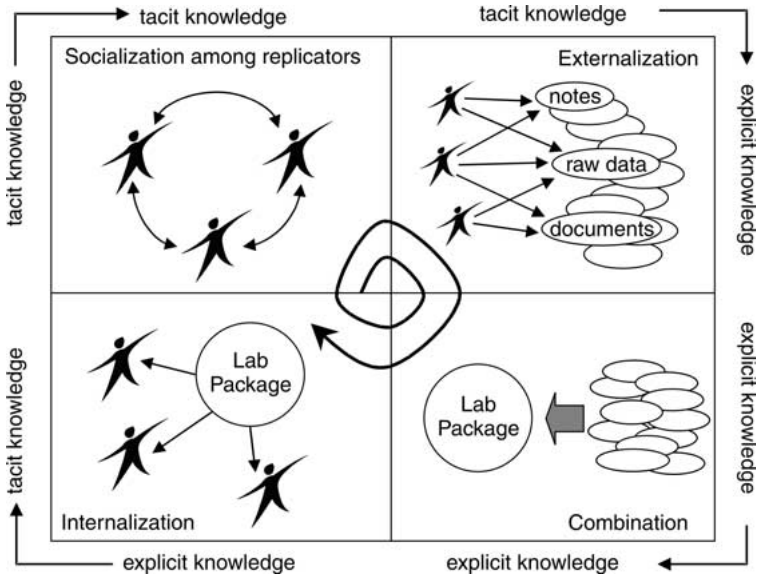

Figure 2. Experimentation knowledge-sharing model (EKSM).

transference of tacit knowledge, knowledge that is not—and could not be—written anywhere in the lab package.

Figure 2 shows an adaptation of Nonaka–Takeuchi model in the context of software engineering experimentation, in what we call the experimentation knowledge-sharing model (EKSM). The four phases of the original model map to: sharing tacit knowledge through socialization among replicators, making tacit knowledge explicit through lab packages, improving explicit knowledge through lab package evolution, and internalizing experimental knowledge by using the lab packages.

Socialization involves the discussion of potential experimental designs and their threats to validity, variables, procedures, data collection methods, and data analysis procedures. It assumes experimenters are using a common terminology and are communicating common cultural themes. Externalization is the recording of notes, tips, experimental design, raw data, analyses, and results. It involves explicit and implicit choices of what should be written down. It is in these implicit choices that a lot of important tacit information is lost. Combination is the decision of what should go into the lab package for other experimenters to use. This process often involves more externalization, such as changes to the process that occurred during the execution of the experiment. Internalization involves the reading, abstraction, interpretation, and execution of what was in the lab package. The execution process often opens questions and points to holes in the lab package.

The execution of experimental replications takes us through all four steps of the EKSM: internalization, socialization, externalization, and combination. The cycle is continuous but not as sequential as it sounds. Many times there are iterations among the steps. The complexity and sophistication of each of those steps evolve with time

and are dependent both on the proximity of the researchers and the mechanisms used to share knowledge among them. The next discusses how the EKSM steps were executed by the Readers' project through the running of two experimental replications. Section 5 discusses the knowledge sharing mechanisms we have been using in these steps.

4. Using Two Experimental Replications to Illustrate the EKSM

The two replications discussed here are aimed at replicating previous experiments on "reading techniques" for human-based review of software requirements for finding defects. The replicated experiment is called the "PBR Experiment." It is being used to evaluate perspective based reading (PBR), a technique developed by the Experimental Software Engineering Group at the University of Maryland (Basili et al., 1996)—one of the research partners in the Readers' Project. PBR aims to improve the effectiveness of software requirements inspections by providing inspectors with procedural guidance for finding defects during their individual review of a requirements document. PBR requires the reader to first create an abstraction of the product, and then to answer questions based on the analysis of this abstraction from a particular perspective that he assumes. The readers are asked to assume the perspective of some stakeholder of the requirements document while they do the review; a "basic set" of perspectives including a software designer (D), a tester (T), and an end-user (U) has been identified and was used for this review. For each perspective, the readers address a set of questions that allow them to discover the defects in the software artifacts (e.g. the questions for the tester perspective lead the reader to discover those requirement defects that would be found by testing the final product). The set of questions are driven by the taxonomy of requirements defects.

The PBR experiments are useful for illustrating the EKSM because there is a sizeable associated body of knowledge (Basili et al., 1996, 1999; Ciolkowski et al., 1997; Laitenberger et al., 2001). We will use the replications to address the following issues related to our high-level goals:

- *Transfer of know-how*. Experimental know-how had to be transferred from the original experimenters to the replicators, and experimental results and experiences had to be transferred back from replicators to the original experimenters, through internalization and socialization.

- *Packaging the experimental artifacts*. A lab package already existed for supporting the PBR Experiment. The replications provided a chance to evaluate whether the package contained sufficient materials and was organized appropriately to transfer the necessary information for executing the experiment (i.e., if they were good for internalization). The replication executions produced material through externalization. More importantly, the Readers' Project gave the replicators and the original experimenters the opportunity to jointly examine how to package and further evolve the required materials through socialization and combination.

These efforts served to test our mode of collaboration, identify its limitations, and address the limitations by adopting new knowledge transfer mechanisms in the Readers' Project.

4.1. The PBR Experiments

The goal of this experiment was to test the following hypothesis:

H0: There is no difference between subjects using PBR and subjects using the standard approach (checklist) with respect to individual effectiveness and efficiency.

Ha: Subjects using PBR and subjects using checklist differ with respect to individual effectiveness and efficiency.

Effectiveness was measured as the percentage of defects found in a requirements document with seeded defects. Efficiency was defined as the number of defects found per hour of time spent on the inspection. Given the relatively small number of subjects, we used an alpha-level of 0.10.

The original experiment took place at the University of Maryland in the US, with professional software developers from NASA's Goddard Space Flight Center. The two replications reported here occurred in Brazil at the University of São Paulo and Federal University of São Carlos respectively. Both replications were run with 18 undergraduate students who were relatively inexperienced in software development (slightly more than one year of classroom experience on average).

The basic design of the study was a within-subjects comparison, in which subjects first used a checklist approach to review a requirements document, were trained in PBR, and then applied PBR to review a different document.

Two different requirements documents were used, both containing seeded defects. The order of the documents was assigned randomly to subjects, so that half of the subjects performed the checklist review on Document A and the PBR review on Document B, while the other half did the Checklist on Document B and PBR on Document A. The effectiveness of each review was measured as the percentage of the seeded defects uncovered during the review. Thus for each subject, the effectiveness during each of the two reviews could be compared to discover whether there was any net improvement due to PBR.

4.2. Running Two Replications as Part of a Series

The two replications use the same requirements documents. Defects reported by subjects are either on the list of defects from the original experiment, considered as new defects, or false positives. Both Checklist and PBR were applied in sessions of 01 h 45 min each, though most subjects finished the reading activity before the

allocated time had elapsed. In replication 1 (R1), subjects were not asked to register the elapsed time between starting and finding each defect. In replication 2 (R2), they were asked to register this information to allow further data analysis such as regarding the technique's learning curve.

In R1, 18 undergraduate students from the Software Engineering course at University of São Paulo at São Carlos carried out the experiment, which consisted of the following steps:

1. Subjects filled out the Consent and the Analyst Survey Forms and were assigned to one of two groups.

2. Subjects applied the techniques as follows: On the first day, all subjects were trained in the baseline (Checklist) method. Subjects in Group 1 then reviewed the ATM Requirements Document and subjects in Group 2 reviewed the PG Requirements Document. On the second day, each subject was trained in one of the three PBR perspectives. Subjects then reviewed the other requirements document, i.e., Group 1 reviewed PG and Group 2 reviewed ATM.

3. Data was collected and results analyzed by experimenters.

4. Subjects received feedback from experimenters.

R2 was quite similar to R1: 18 undergraduate students from the Software Engineering course at the Federal University of São Carlos conducted the experiment, consisting of the following steps:

1. Subjects filled out the Consent and the Analyst Survey Forms.

2. The experiment follows the same experimental design of the previous ones, and was divided in four half-day periods. Subjects applied the techniques as follows: On the first half-day, all subjects were given an overview on inspection techniques and trained in the baseline (Checklist) method. In the second half-day subjects from Group 1 reviewed the ATM Requirements Document and subjects in Group 2 reviewed the PG Requirements Document. On the third half-day, each subject was trained in one of three PBR perspectives. Then, in the fourth half-day, subjects reviewed the other requirements document, i.e., Group 1 reviewed PG and Group 2 reviewed ATM.

3. Data collection and analysis of the results by the experimenters.

4. Subjects received feedback from the experimenters.

Different persons trained subjects in both replications, and the trainer in R2 was more experienced. Although in both cases it was their first training session, the trainer for R2 had closely followed the procedures in R1. Consequently, we

believe that training sessions in R2 were as good as those of R1, if not better. The Defect Collection Form was slightly expanded to collect two additional attributes for each reported defect, namely the number of the requirement in which the defect was found, and the time of defect identification. Reporting those attributes required little extra effort from the subjects and improved the data analysis.

4.3. Results Obtained

In this section, we provide a brief, high-level comparison of results obtained in the two replications. Our goal here is not to provide a detailed discussion of the experimental results but rather to understand the overall results observed from each replication, and the consistency of results across studies. Detailed discussion of these replications and their results will appear in forthcoming papers.

Figure 3 summarizes the effectiveness of the subjects for both replications. The results from R1 and R2 are similar for the ATM document but differ for the PG document. For the ATM document, in both R1 and R2 the subjects using the PBR technique found a higher percentage of the defects on average than the subjects using the checklist. On the other hand, for the PG document, in R1 the subjects using the checklist found a higher percentage of the defects on average than the subjects using PBR, while in R2 the subjects using PBR found a higher percentage of the defects than the subjects using the checklist. In the original study, the subjects using PBR found more defects than the subjects using the checklist, on both documents. Therefore, the results from R1 are in partial support of the results from the original study, while the results from R2 fully support the results of the original study.

Because of the additional tacit knowledge that was made explicit during R1, the results of R2 appear to be more comparable to those of the original study. Further replications are ongoing to determine the cause of the differences between the two studies, whether they be cultural factors, differences in subjects's skill levels, differences in running of the experiment, or other changes.

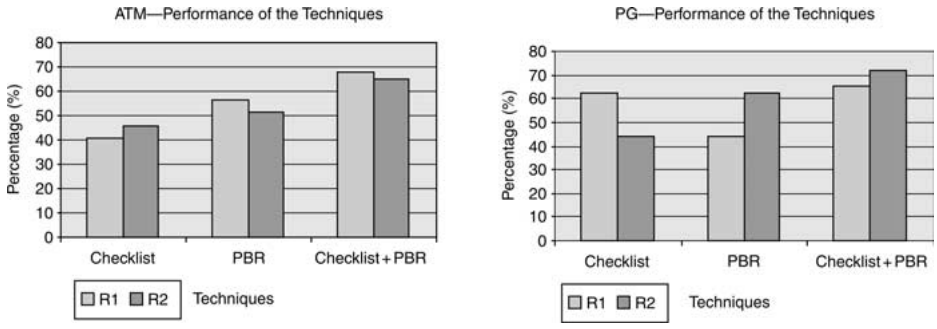

Figure 3. Inspection performance on each document, for both replications.

4.4. Instantiating the EKSM during the Replications

The activities required to conduct the replications described above fit the phases of the EKSM (described in Figure 2).

The first step in R1 was for the replicators to gain access to the laboratory package with its associated artifacts. That was an easy task due to the framework of the Readers' Project; however, one should keep in mind that this framework is not always present. This process may involve negotiations regarding artifacts and results ownership between the package providers and the replicators.

A difficulty that the replicators had was to *assemble a complete and consistent lab package for the replication*, i.e., to assemble the right set of explicit knowledge for executing the replication. The original lab package consisted of several artifacts that, due to their use in other replications, had evolved over time. The identification of compatible and/or consistent artifacts was not an easy task. It was an exercise of combining explicit knowledge done jointly by the replicators and original experimenters. The artifact repository at the University of Maryland (the original experimenters) was kept in a simple file system structure. The replicators had to identify, for example, the best list of defects for the most current requirements specifications. This *version control and configuration problem* arose due to the growth of the number of artifacts available in the UMD experience base. It indicates that we need more sophisticated tools to keep track of experimental artifacts, which in a sense is a good sign. It shows that we have a growing body of knowledge in software engineering experimentation.

After gathering all the artifacts, the replicators still had to adapt some of them. They had to include, for example, questions regarding English language expertise in the Subject Characterization Questionnaire. This adaptation always happens when there are variations between the original and the replicated experimental settings. In this case, the *replications had different cultural settings*—Brazilians have different levels of expertise on the artifact language (English), a problem that was not present in the original US experiments. Other types of variations may appear in experimental replications of this kind, for example, variations between replications done in academic and industrial settings.

Because of the level of effort involved in running even a replicated experiment, the replicators were not willing to undertake a full experiment without testing the material and concepts in their own environment. As a result, they decided to *run a pilot study to better understand how the experimental process should be conducted*, paying attention especially to experimental mechanics such as the timing issues, tasks to be executed and documents to be delivered to the subjects. Although the pilot study also required an investment of effort and resources, it was considered appropriate and necessary since this was the first time this experiment was undertaken outside the Maryland umbrella and it seemed to afford the best opportunity to identify and master the tacit knowledge issues. The pilot study was intended to ensure the quality and conformance of the experimental process.

The *pilot study was very useful for identifying tacit knowledge issues*. For example, the replicators could not really understand why the same amount of training time

was specified for both the checklist and PBR approaches, since they provide different levels of detail and require different levels of background knowledge. For each perspective of PBR the operational aspects have to be taught in addition to the understanding of the questions and of the requirements defect taxonomy. The replicators made some adjustments to the training time but kept it equal for both techniques. It is important to point out that this aspect should be seen as a threat to validity, because it probably was not fair to use equal training time for two techniques of varying complexity. This point was shared with the original experimenters and should be addressed in further experiments.

Based on our experience, we recommend including in any replication process a pilot study aiming at mastering the underlying concepts and tacit knowledge and also providing a mechanism to assess the process conformance before any significant replication effort is undertaken. Aside from identifying missing knowledge, pilot studies also assists in the process of internalization and socialization; i.e., the replicating researchers are afforded the chance to better understand the relevant knowledge by running their own study and to share their new understanding more easily with others of the co-located group.

One additional important note on the first replication is that the subjects found several new defects in the seeded documents in addition to those already documented in previous experiments. Thus, in addition to the existing version control problem, there was an *update to the defect lists* occurring as the project was running. This update required a decision as to whether to re-analyze previous experimental data to ensure consistency among all results. As a result of this new knowledge, a process of externalization was begun, by which the new tacit knowledge that had been discovered by the group had to be made explicit, in the form of updates to the package, in order to be available for future replications (see Figure 2).

Another example of externalization of useful tacit knowledge was the *creation of a list of frequent false positives*. The false positives are those defects reported by the inspectors that are not real defects; they are, normally, outside of the scope of the requirements documents. Because of the lack of standards and the degree of variation in scope of requirements documents in industry, it is a difficult task to define which issues are real defects and which are not. In the replications, there were issues in the inspected documents that were frequently reported by subjects as defects but not accepted by the experimenters. The frequent false positive list avoided a lot of data interpretation (re)work.

A decision process for when defects and false positives should be added to their respective lists was adopted by the project participants. It should be pointed out that updating the lists of defects and false positives are examples of feedback loops where information originating from the replicators is fed back to original experimenters leading to knowledge evolution.

R2 was much easier to set up and execute because much of the tacit knowledge needed was available locally. An important issue that it raised was on *how to combine and analyze data from different experiments*. Much attention was given to the problem of subjective measures such as those collected in the subject characterization and feedback questionnaires. We concluded that the subject profile introduces

several key intervening variables (e.g. experience with technique, language expertise) in the experiment that should be better measured. Collecting information on each defect reporting time was discussed among the partners and considered useful for further data meta-analysis.

It is important to point out that analyzing the results of such experiments is not an easy task. There are many ways to analyze the information from a typical experiment, and it is often the case that simply testing the experimental hypotheses neglects a significant percentage of the data collected. For example, once a technology has been evaluated by comparison to another technology, it is necessary to evaluate the data from different points of view to better understand the reasons for the success or failure of the technology. For instance, given a specific defect, who are the subjects that found it? Given a specific perspective, which defects were more likely found by using it rather than another? For a specific class of defects, which subjects did better? For this reason we recommend that *hypothesis testing* be complemented by a discovery-driven approach, with the aim of *hypothesis generation* for guiding further studies. This process, comparable to data mining over the data collected by an experiment, is again difficult to do in isolation and benefits greatly from a close collaboration where hypotheses can be brainstormed, debated, and the data quickly checked. In other words, experimental evolution is frequently dictated by extended data analyses and revisiting data from previous studies, in the same vein as proposed by Carver (2003). Those are issues that depend heavily on effective management of experimental knowledge, both explicit and tacit knowledge. This requires an environment in which socialization and internalization of knowledge is facilitated.

5. Knowledge Sharing Structures in the Readers Project

Based on our experiences with experimental replications in the Readers' Project, we have found that there is a need for a wide range of mechanisms to conduct this collaboration and effectively share knowledge among the researchers involved. These mechanisms are needed to instantiate the experimentation knowledge sharing model depicted in Figure 2. We have used e-mail, web portals, knowledge repositories, workshops, and eWorkshops to share knowledge among the participants of the Readers' Project. These mechanisms have evolved in sophistication (although not necessarily in complexity) over time. The following subsections discuss each of these mechanisms in turn.

5.1. Workshops

The primary mechanism for information exchange in the Readers' Project is a series of semi-annual workshops, alternating in location between the two countries of the participants. The workshops usually have two or three representatives of each participating organization, with the exception of the host organization, which

contributes more participants. Participants from the host may include graduate students and sometimes invited researchers. With this structure, the workshop audiences have ranged from 12 to 25 people. The workshops have a well defined agenda that plan for three or four days of very focused work.

The 1st Workshop (October 1999, Florianópolis, Brazil) focused on summarizing each organization's contributed expertise. This was followed by discussions on the project agenda, goals and expected results. Lastly, we choose experimental designs and possible artifacts for upcoming experiments.

The 2nd Workshop (March 2000, São Carlos, Brazil) focused on giving the go-ahead for the first experimental replications—reading for defect detection in requirements, and comparing reading and testing for code. The experimental designs and artifacts were reviewed jointly by all partners. We also discussed how to evolve reading and testing techniques.

The 3rd Workshop (January 2001, College Park, USA) focused on discussing the first experimental results. We started addressing lab-package quality factors and discussed future experiments and work synchronization. Lastly, we planned for future workshops, effective means of interaction, and publication of results.

The 4th Workshop (June 2001, College Park, USA) focused on discussing new experimental results but also on artifact evolution. We also discussed alternative means of data analysis and data exploration, and we adopted the CeBASE experience base (Basili et al., 2001a) as the knowledge repository for the project (see Section 5.3).

The 5th Workshop (January 2002, Salvador, Brazil) discussed several new experimental results, but it also focused heavily on experience packaging. We discussed methodologies and criteria for artifacts evolution and mechanisms for controlling and disseminating artifacts. We also discussed knowledge management tool support, especially for group communication and cooperation across different locations (see Sections 5.2, 5.3, and 5.4).

The 6th Workshop (July 2002, College Park, USA) discussed further experimental results, but it focused heavily discussing how experiments evolve. We discussed how to build grounded theory and how hypotheses have evolved over experiments (Carver, 2003). We also started discussions on knowledge sharing and tacit knowledge problems (Shull et al., 2002).

The 7th Workshop (Rio de Janeiro, Brazil) focused little on actual experimental results. We focused instead on data meta-analysis across several experiments and the requirements it imposes on data collection, data quality, and information packaging. A theory on experimental evolution based on grounded theory was presented (Carver, 2003). We spent a good portion of the workshop discussing knowledge management and associated tool support. An architecture for a Readers' Project experience and data base was adopted. The FC-MD eWorkshop tool was officially adopted to complement the regular workshops (Basili et al., 2001a).

The workshops were a key mechanism to execute the socialization and composition steps of the EKSM depicted in Figure 2. Reading back through the workshop summaries, one can observe that the discussions moved from specific knowledge transfer issues, such as how to execute an experiment and reporting the

results of an experiment, to broader knowledge evolution and sharing issues, such as experiment improvement, how to combine data from different experiments, and how to address the tacit knowledge problem on our experiments. The physical presence was key to address the toughest questions related to tacit knowledge transfer at the EKSM's socialization step and package evolution at the EKSM's knowledge combination step.

5.2. Email-and Web Portals

While the workshops have been extremely effective for knowledge sharing, we are unable (due to time constraints, travel costs, and so on) to have them more often than twice per year. We have found that augmenting these workshops with other mechanisms was a must. That is, there is a need for a wide range of mechanisms to permit sharing knowledge on issues such as experimental design and experiment process execution details as necessary, sharing ideas and findings and correcting problems as fast as possible.

We have found that augmenting the workshops with very close e-mail contact was useful for answers to focused questions and peer-to-peer information exchange. In this way, e-mail facilitated socialization among the researchers and nicely complemented the workshops, in the sense that they allowed frequent (but limited) transfers of knowledge.

We built web pages and, later, a web portal (http://www.labes.icmc.usp.br/readers) as a common repository for lab packages, workshop materials, and documents associated with the project. From the perspective of the EKSM model, the portal is useful for helping knowledge externalization and internalization. It is important to notice, however, that tools such as e-mail and web portals are insufficient for fulfilling all the needs of undertaking replications. E-mail correspondence is sufficient for answers to focused questions but does not facilitate the kinds of observations necessary to identify the multiplicity of issues arising from the complexity and interrelated nature of experimental design. Web portals are useful to store raw material and documents, but are insufficient to capture the relationships among the various objects involved in software engineering replications (artifacts, questionnaires, subjects, raw data, etc.), support their instantiation (using for example templates and meta-models), and their evolution, configuration and version control.

5.3. Knowledge Repository

As we are now much more focused on the meta-data analysis, experiment improvement, knowledge evolution and sharing, our efforts in these directions have demanded new technological support for managing experiments and the knowledge produced by them. The project has adopted the CeBase repository (www.cebase.org) at the Fraunhofer Center Maryland as part of the infrastructure to

manage experience packages (Basili et al., 2001a). The repository is based on Hyperwave, a knowledge management support tool (Hyperwave, 2003), and the same infrastructure is adopted by the partners in the project. The current infrastructure supports document management, material ownership, and roles.

Based on this experience, a more complete infrastructure is now being developed. It explores the narrative package model described by Conradi et al. (2001) and the experimentation process of Wholin et al. (2000) to define an improved packaging process approach (Amaral and Travassos, 2003). In our approach, packaging activities are intended to be accomplished throughout the experimentation process rather than just as the last activity of the experimentation process. Both experimentation and packaging processes definitions and the graphical document package model representation (UML based description) have been used to guide the implementation of specific packaging facilities in Hyperwave. A more ambitious architecture is being built to support experimentation knowledge management and experiment execution. This architecture will provide an experience base and a standard layer for integrating tools supporting experimental execution, meta-data analysis, and knowledge sharing. Our final goal is to use it to efficiently guide users through the tasks of knowledge externalization and internalization of the EKSM depicted in Figure 2.

5.4. eWorkshops and Local Workshops

Recently, we have started holding local workshops. These workshops are a consequence of a greater integration between the groups, especially with respect to building knowledge management infrastructures, and of the growth of the number of people involved in the project. These workshops involve fewer participants than the semi-annual workshops and their purpose is three fold: (1) sharing knowledge at smaller scale; (2) coordinating work groups for tasks that involve only two or three groups of researchers; and (2) preparing for the semi-annual workshops. From the perspective of the EKSM, they work as an auxiliary means of socialization among the project participants.

Besides the semi-annual workshops, we have adopted electronic workshops or ''eWorkshops'' for more focused and shorter knowledge sharing sessions (Basili et al., 2001a). The eWorkshops are held through the web using a tool developed by the Fraunhofer center (FC-MD) that integrates chat, whiteboard, and voting mechanisms. The eWorkshop process is well defined. It has established roles for participants, observers, lead discussants, and moderators. We prepare a well-defined and focused agenda ahead of the eWorkshop. The eWorkshop session itself involves all the geographically distributed partners and lasts for up to two hours. It embodies a very intense exchange of messages through the chat mechanisms, summarizing decisions on the whiteboard, and calling for voting when a decision is necessary. We run a few of those sessions each semester in between the regular workshops. It has been proven to be an excellent socialization tool.

5.5. *A Final Word on Collaboration Mechanisms*

Besides the collaboration structures described here, we have considered a wide range of mechanisms to permit sharing knowledge. However, we adopt knowledge sharing mechanisms only when we firmly believe that they will be useful for the main goal of the project, that is, to build a body of knowledge on reading and testing techniques through experimentation.

6. Importance of Knowledge Sharing Structures to Experimental Software Engineering

To facilitate experiment replications in software engineering, it becomes necessary to use effective mechanisms for sharing knowledge among the original experimenters and the replicators. This knowledge transfer task is not as simple and straightforward as one might expect. Issues such as addressing the tacit knowledge problem, formalizing an experiment into a lab package, and evolving it over time are heavily dependent on how well knowledge is shared.

6.1. *Communicating Tacit Knowledge*

In this paper, we have used a model of knowledge transfer (EKSM) to organize and describe the problem areas and weak points in experimental replications in empirical software engineering. As can be noted from the previous discussion, the most significant problems were encountered in the areas of internalization and socialization because of difficulties in transferring and absorbing tacit knowledge. Key to addressing these areas is the use of knowledge sharing mechanisms such as the ones we have used in this project.

The tacit knowledge problem exists in part because it is difficult for the original experimenter to know enough about what the replicators are going to need to make all of the relevant knowledge explicit. Better understanding these needs is one positive outcome of doing the replications. For instance, we found out that it was important to detail the procedures for the replications, as much as possible, determining timing and documents to be delivered at each step of the experimental process for both training and execution phases.

Based on our experience, we have also found out that an effective way of communicating knowledge about an experiment is to explicitly include in the lab package more than just a description of what is to be done in the experiment. By also including a list of issues that are invalid or outside the scope, along with rationales for these decisions, the experiment can be delimited more clearly. This is an example of a type of tacit knowledge that is usually overlooked in the externalization step, but which can have large benefits when made explicit. For example, another of the major contributions of this PBR replication was the creation of a list of "frequent false

positives,'' that is, a list of issues in the documents inspected that were frequently reported by subjects as defects but not accepted by the experimenters.

6.2. Evolving the Lab Package

We believe that requirements documents for experimentation should ideally have two versions, a seeded version and a golden version, providing another mechanism for making additional tacit knowledge explicit. The golden version should be a correct or nearly correct version. Its usefulness would be two-fold: (1) to help build an oracle version and an error list through error seeding; and (2) to ease the process of analyzing the errors reported during the experiment. The oracle version is the version used in the experiments that has been seeded with defects. We also believe that a body of experimenters should decide on the evolution of the oracle version and its list of defects and false positives. This body of experimenters should be made up of experienced experimenters (Basili et al., 2001a,b; Mendonça et al., 2001). This last issue signals once again the need for using adequate communication and socialization mechanisms among experimenters and replicators.

As a concrete result of our replications, the laboratory package, which should serve as a baseline for further experimental replications, has evolved. Changes include the definition of an explicit process for training sessions and defect analysis, the elaboration of a list of defects and another one of frequent false positives, and the evolution of the analyst and feedback questionnaires.

6.3. Package Quality Goals

Based on experiences, we have also outlined a series of goals that an effective lab package should be able to support, aiming at contributing to the definition of an experimental knowledge repository in the same vein as Conradi et al. (2001).

These quality goals can be used as an informal evaluation checklist to assess the design of specific lab packages. As we continue to work with lab packages, we are trying to find useful metrics that can more accurately assess packages. The quality of a lab package can be assessed by how well it supports the following goals, many of which are involved with knowledge sharing effectiveness:

- Access to experimental artifacts.
- Adequate and complete training materials (Complete materials include aspects such as the necessary background to comprehend the technology and take language issues into account).
- Accurate estimates for the time required for subjects to execute the experiment.

- Presence of oracles and golden artifacts (materials against which the quality of the systems being developed using the technology can be measured).
- Ease of package evolution and experimental feedback.
- Clear description of communications channels by which the original experimenters and, if possible, related experimental community can be contacted.
- Complete descriptions of:
  - The analysis and goal of the experiment (so that other researchers can evaluate the feasibility of the experiment for their environment).
  - The experimental design, including threats to validity and strengths of experiment (so that other researchers can make changes to the design), along with the decision history (justifying decisions and reflecting on what works as well as what does not).
  - The context(s) in which the experiment was run.
  - The process to run the experiment.

7. Conclusions and Next Steps

Although the replication of experiments is a necessary component of research in software engineering and crucial to our ability as a field to understand fundamental principles of software development, running effective replications is not straightforward. The need for knowledge transfer is one important reason why this is so, due to the sensitivity of experimental designs involving human subjects, where small variations in the execution of the experiment can have large effects on results. Stand-alone lab packages cannot be the entire answer to facilitating replications. We provided several examples of such difficulties in replicated experiments, both within the Readers' Project and without, to further motivate this argument. We wish to again emphasize that these difficulties are not the fault of inattentive researchers, but have occurred in spite of having experienced researchers as both the original experimenters and replicators, extensive preparation effort on both sides, mature materials, and lab packages explicitly designed to promote replication.

In a previous paper, we have argued that lab packages must be augmented by a replication process and attention to process conformance within the experiment (Shull et al., 2002). In this paper, we argue that knowledge sharing among the original experimenters and replicators is a crucial aspect of ensuring process conformance between experiments, adressing the tacit knowledge problem, formalizing an experiment into a lab package, and evolving it over time. In this paper, we have instantiated the Nonaka–Takeuchi Knowledge Sharing Model for

software engineering experimentation and laid out the knowledge sharing mechanisms we have been using in the Readers' Project. In further work, we will be investigating issues such as: (1) modeling and enacting experimentation and packaging processes; (2) formalizing an experimentation improvement meta-model; and (3) building an experimentation knowledge repository.

Acknowledgments

We would like to thank the members of the Readers' Project team for their comments during the project workshops, which helped improve this paper. We would also like to thank the subjects who took part in the studies described here.

This work has been accomplished under the financial support of CNPq and NSF grants CCR9900307 and CCR0086078 (establishing the Center for Empirically Based Software Engineering).

Notes

1. To avoid this danger we have deliberately avoided giving names or references.

References

Agresti, W. 2000. Knowledge management. *Advances in Computers* 53: 171–283.

Amaral, E. A. G. G., and Travassos, G. H. 2003. A package model for software engineering experiments. *Proc. of the 2003 ACM-IEEE International Symposium on Empirical Software Engineering ISESE (ISESE 2003)*. Rome. (To appear—poster session).

Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Soerumgaard, S., and Zelkowitz, M. V. 1996. The empirical investigation of perspective-based reading. *Empirical Software Engineering* 1(2): 133–164.

Basili, V. R., Shull, F., and Lanubile, F. 1999. Building knowledge through families of experiments. *IEEE Trans. Software Engineering* 25(4): 456–473.

Basili, V. R., Tesoriero, R., Costa, P., Lindvall, M., Rus, I., Shull, F., and Zelkowitz, M. V. 2001. Building an experience base for software engineering: A report on the first CeBASE eWorkshop. *PROFES (Product Focused Software Process Improvement)*. Kaiserslautern, Germnay, 110–125.

Basili, V. R., Lindvall, M., and Costa, P. 2000. Implementing the experience factory concepts as a set of experience bases. *Proc. The 13th International Conference on Software Engineering & Knowledge Engineering*. Buenos Aires, 102–109.

Brooks, A., Daly, J., Miller, J., Roper, M., and Wood, M. 1996. Replication of experimental results in software engineering. ISERN Technical Report ISERN-96–10.

Carver, J. 2003. The impact of background and experience on software inspections. PhD Thesis, Dept. of Computer Science, University of Maryland, April, 2003. (Also available as University of Maryland Department of Computer Science Technical Report CS-TR-#4476.)

Ciolkowski, C., Differding, C., Laitenberger, O., and Muench, J. 1997. Empirical investigation of perspective-based reading: A replicated experiment. ISERN Technical Report ISERN-97–13.

Conradi, R., Basili, V. R., Carver, J., Shull, F., and Travassos, G. H. 2001. A pragmatic documents standard for an experience library: Roles, documents, contents and structure. University of Maryland Technical Report CS-TR-4235.

Dixon, N. M. 2000. *Common Knowledge: How Companies Thrive by Sharing what They Know*. Boston, MA: Harvard Business School Press.

Goth, G. 2001. New center will help software development ''grow up''. *IEEE Software* 18(3): 99–102.

Johnson, P. 1996. BRIE: The benchmark inspection experiment. Department of Information and Computer Sciences, University of Hawaii, Technical Report csdl-93–13.

Laitenberger, O., El Emam, K., and Harbich, T. 2001. An internally replicated quasi-experimental comparison of checklist and perspective-based reading of code documents. *IEEE Transactions on Software Engineering* 27(5): 387–421.

Lindvall, M., Rus, I., and Sinha, S. 2002. Technology support for knowledge management. *Proc. 4th International Workshop on Learning Software Organizations (LSO '02)*, Chicago, Illinois.

Lott, C., and Rombach, D. 1996. Repeatable software engineering experiments for comparing defect-detection techniques. *Empirical Software Engineering An International Journal* 1(3): 241–277.

Hyperwave, 2003. http://www.haup.org.

Mendonça, M. G. and Sunderhaft N. L. 1999. A state of the art report: Mining software engineering data. Rome, NY: U.S. Department of Defense (DoD) Data & Analysis Center for Software, also available at http://www.dacs.dtic.mil/techs/datamining/.

Mendonça Neto, M. G., Seaman, C., Basili, V. R., and Kim, Y. M. 2001. A prototype experience management system for a software consulting organization. *Thirteenth International Conference on Software Engineering and Knowledge Engineering*. Buenos Aires, 29–36.

Miller, J. 2000. Applying meta-analytical procedures to software engineering experiments. *Journal of Systems and Software* 54: 29–39.

Nonaka, I., and Takeuchi, H. 1995. *The Knowledge Creating Company*. Oxford, UK: Oxford University Press.

Rus, I., Lindvall, M., and Sinha, S. 2001. Knowledge management in software engineering. The Data & Analysis Center for Software (DACS) State-of-the-Art-Report.

Shull, F., Carver, J., and Travassos, G. H. 2001. An empirical methodology for introducing software processes. *Proc. 8th European Software Engineering Conference*. Vienna, Austria, 288–296.

Shull, F., Basili, V. R., Carver, J., Maldonado, J. C., Travassos, G. H., Mendonça Neto, M. G., and Fabbri, S. C. P. F. 2001. Replicating software engineering experiments: Addressing the tacit knowledge problem. *Proc. of the 2002 International Symposium on Empirical Software Engineering (ISESE'2002)*. Nara, Japan, 7–16.

Wholin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslen, A. 2000. *Experimentation in Software Engineering: An Introduction*. Massachusetts: Kluwer Academic Publishers.

Wiig, K. 1999. Comprehensive knowledge management. Knowledge Research Institute, Inc. Working Paper KRI #1999–4 Revision 2.

Forrest Shull received his doctorate from the University of Maryland College Park in 1998 for work on software inspection techniques. Since then he has published numerous articles and book chapters on the topic, given tutorials at conferences such as ICSE, consulted with private companies and government agencies, helped introduce improved inspection techniques to the software engineering curriculum at universities, and performed applied research to further refine the techniques. With extensive experience in the area of empirical assessments of software development technologies, Dr. Shull is a scientist at the Fraunhofer Center for Experimental Software Engineering in Maryland, and also serves as a principal member of the NSF-funded national Center for Empirically Based Software Engineering (CeBASE).

Manoel G. Mendonca received his PhD in computer science from the University of Maryland in 1997. He also holds a MSc in computer engineering from the State University of Campinas (1990), and a BSEE in electrical engineering from the Federal University of Bahia (1986), both in Brazil. Dr. Mendonca was a visiting scientist and was awarded a doctoral fellowship from the IBM Toronto Laboratory's center for Advanced Studies. Between 1997 and 2000, he worked as a Faculty Research Associate at the University of Maryland and as a scientist at the Fraunhofer Center for Experimental Software Engineering in Maryland. He is now a professor at Salvador University (UNIFACS) in Brazil. His main research interests are on data mining, experimental software engineering, and knowledge management supporting systems.

Victor R. Basili is a professor of computer science at the University of Maryland, College Park, the executive director of the Fraunhofer Center—Maryland, and one of the founders and principals in the Software Engineering Laboratory (SEL). He works on measuring, evaluating, and improving the software development process and product and has consulted for many organizations, including AT&T, Boeing, Daimler-Chrysler, Ericsson, FAA, GE, GTE, IBM, Lucent, MCC, Motorola, NRL, NSWC, and NASA. He is a recipient of a 1989 NASA Group Achievement Award, a 1990 NASA/GSFC Productivity Improvement and Quality Enhancement Award, the 1997 Award for Outstanding Achievement in Mathematics and Computer Science by the Washington Academy of Sciences, and the 2000 Outstanding Research Award from ACM SIGSOFT. He has authored more than 150 journal and refereed conference papers, has served as editor-in-chief of the IEEE Transactions on Software Engineering, and as program chair and general chair of the Sixth and 15th International Conference on Software Engineering, respectively. He is an IEEE and ACM fellow.

Jeffrey Carver is a Faculty Research Associate at the University of Maryland. He received his PhD from the University of Maryland in 2003. His PhD thesis was entitled "The Impact of Background and

Experience on Software Inspections.'' His research interests include: Empirical Software Engineering, Software Inspections, Software Metrics, and Qualitative Methods. In addition to refining the qualitative research method developed in his thesis, his current work is focusing on software development for High Performance Computers.

José Carlos Maldonado received his BS in Electrical Engineering/Electronics in 1978 from the University of São Paulo (USP), Brazil, his MS in Telecommunications/Digital Systems in 1983 from the National Space Research Institute (INPE), Brazil, and his DS degree in Electrical Engineering/Automation and Control in 1991 from the University of Campinas (UNICAMP), Brazil. He worked at INPE from 1979 up to 1985 as a researcher. In 1985 he joined the Computer Science Department of the Institute of Mathematics and Computer Science and at the University of São Paulo (ICMC-USP) where he is currently the head of the department. His research interests are in the areas of Software Engineering, Software Testing and Validation and Software Quality and Software Testing. He is the Vice-president of the SBC—Brazilian Computer Society (2003–2005).

Sandra Fabbri is an assistant professor in the Department of Computer Science at Federal University of São Carlos, São Paulo, Brazil. She has a degree in Computer Science from Unicamp—University of Campinas. She received a MSc in Information System area and a PhD in Software Engineering area from USP—University of São Paulo. Her research interests include Software Testing, Formal Specification, Software Quality, Requirement Engineering and Experimental Software Engineering.

Guilherme H. Travassos is an Associate Professor of Computer Science at COPPE—Federal University of Rio de Janeiro—Brazil. He received his doctorate degree from COPPE/UFRJ in 1994. He is a former faculty researcher (1998/2000) with the Experimental Software Engineering Group at the Department of

Computer Science, University of Maryland. His current research interests include empirical software engineering, software quality, software engineering environments and process improvement. Dr. Travassos has authored several journal, refereed conference papers and book chapters. He has industrial experience concerned with leading software development projects. Dr. Travassos is member of ISERN, ACM and Brazilian Computer Society.

Maria Cristina Ferreira de Oliveira received the BSc in Computer Science from the University of São Paulo, Brazil, in 1985, and the PhD degree in Electronic Engineering from the University of Wales, Bangor, in 1990. She is an Associate Professor at the Computer Science and Statistics Department of the Institute of Mathematical Sciences and Computing, at the University of São Paulo, Brazil, where she teaches Data Structures, Computer Graphics and Visualization, since 1990. Her research interests are in Scientific Visualization and Information Visualization, with particular interest in perception issues, mapping techniques, 3D reconstruction, user interaction, visual data exploration and visualization over the Internet. In Scientific Visualization, she has been collaborating with Brazilian research groups in visualization applied to dentistry, medicine and neuroscience to support exploration and analysis tasks. In Information Visualization she is particularly interested in the integration of visualization and data mining techniques with applications in the analysis of data with spatial and temporal attributes. She has been a visiting scholar at the Institute for Visualization and Perception Research at University of Massachusetts, Lowell, in 2000/2001.