# A Framework for Software Engineering Experimental Replications

Manoel G. Mendonça[1], José C. Maldonado[2], Maria C.F. de Oliveira[2], Jeffrey Carver[3], Sandra C.P.F. Fabbri[4], Forrest Shull[5], Guilherme H. Travassos[6], Erika Nina Höhn[2], Victor R. Basili[5,7a]

*[1]Salvador University, [2]University of São Paulo at São Carlos, [3]Mississippi State University, [4]Federal University of São Carlos, [5]Fraunhofer Center Maryland, [6]Federal University of Rio de Janeiro, [7]University of Maryland at College Park*

*mgmn@unifacs.br, {jcmaldon, cristina, hohn}@icmc.usp.br, carver@cse.msstate.edu, sfabbri@dc.ufscar.br, fshull@fc-md.umd.edu, ght@cos.ufrj.br, basili@cs.umd.edu*

## Abstract

*Experimental replications are very important to the advancement of empirical software engineering. Replications are one of the key mechanisms to confirm previous experimental findings. They are also used to transfer experimental knowledge, to train people, and to expand a base of experimental evidence. Unfortunately, experimental replications are difficult endeavors. It is not easy to transfer experimental know-how and experimental findings. Based on our experience, this paper discusses this problem and proposes a Framework for Improving the Replication of Experiments (FIRE). The FIRE addresses knowledge sharing issues both at the intra-group (internal replications) and inter-group (external replications) levels. It encourages coordination of replications in order to facilitate knowledge transfer for lower cost, higher quality replications and more generalizable results.*

## 1. Introduction

Empirical research is fundamental to the evolution of software engineering as a discipline, but according to Kitchenham et al. the current state of empirical research in software engineering (i.e. case studies, surveys, and formal experiments) is less than ideal [22]. Surveys of the software engineering literature show that the majority of the papers contain little or no empirical validation [27,28,30,31]. This lack of empirical work is more disturbing considering that it is difficult to build a usable body of knowledge from individual studies. The results of an isolated experiment are not likely to be directly applicable to every practitioner due to differences in system domains, subject profiles and cultural environments [18]. Replications must be conducted to ensure more robust evidence sets that can support generally applicable conclusions [5]. These replications can be conducted by the same research group (i.e. internal replications) or by different researchers, in different contexts (i.e. external replications).

External replications can be classified according to whether they were run independent of or coordinated with, the original experimenters. Each produces different kinds of knowledge. In this paper we argue that coordinated external replications facilitate the building of knowledge, the evolution of experimental artifacts, and the analysis of results across the replicated studies in a way that independent external replications do not. Coordinated research helps ensure the compatibility of a set of controlled experiments to allow for the production and integration of a significant body of results.

The problem of conducting effective, coordinated replications has been addressed by the Readers Project - a collaborative research effort, sponsored by the Brazilian (CNPq) and American (NSF) national research foundations. The Readers Project used reading techniques (techniques used for defect detection during software document reviews) to experimentally develop, validate, and package an infrastructure to support effective replications [24]. This focus on reviews and the underlying analysis techniques (i.e. the reading techniques) is important because most software artifacts require continual understanding, review, and modification. A large body of studies has demonstrated that reading techniques are effective for improving individual review practices in different domains and

types of inspection, e.g. natural language requirements [3], requirements in formal notation [20], high-level designs [14], code [13,15], and user interfaces [32].

Based on our experience in the Readers Project this paper discusses the problem of experimental replications and proposes a framework for improving the replication of SE experiments (FIRE) The FIRE addresses knowledge sharing issues both at the intra-group (internal replications) and inter-group (external replications) levels. It encourages coordination of replications in order to facilitate knowledge transfer for lower cost, higher quality replications and more generalizable results.

## 2. Experimental Replications in Software Engineering

Empirical software engineering research should be broader than simply conducting single, isolated studies. It should focus on consolidating a body of knowledge about the costs and benefits of techniques that will enhance the understanding of software development processes, and establish novel software development models. Drawing conclusions from single studies in software engineering is inherently dangerous because sample sizes are often small relative to medical or social science studies and there are a large number of possible intervening factors in human subjects experiments, some controllable, some completely uncontrollable.

Miller states that the use of replications is the standard approach to understand and eliminate intervening factors by changing some of these factors to see if the original result is stable [19]. A series of replicated experiments with small to medium sample sizes help safeguard against:

- Low statistical power
- Unknown interactions of other variables with the treatment variable
- Flaws in the design, process and artifacts of the study

The term replication is central to this work, so, it is important to discuss its varied definitions. In experimental software engineering, which uses human subjects, exact replications are not feasible [9]. Hence, it is more correct to describe replication attempts as partial replications [19]. Therefore, a "good" replication can be defined in two ways:

1. The experimenter minimizes the variation between the replication and the original
2. The experimenter consciously changes a small number of factors to either improve the study or increase the external validity of the whole set of studies.

Since portions of the artifacts, experimental design, and protocols are tested during each replication, the experiment is improved over time. It is therefore imperative to run multiple, replicated studies with two goals:

1. Gathering additional data to increase confidence in the original results (i.e., show that the original result holds across multiple environments and subjects)
2. Addressing deeper research questions, for example by controlling for different factors or using different metrics.

Within this context, there are two types of replications, internal and external. An internal replication is conducted by the same set of researchers who conducted the original experiment, while an external replication is conducted by different researchers. Brooks et al. provide an excellent discussion about replications in a software engineering environment, highlighting the need for external replications to provide a strong scientific foundation [8].

Important progress can be achieved by using guidelines and packages to support replications, as shown in prior research [7,16,29]. Although these efforts made significant contributions to establishing guidelines for replications, none explicitly addressed methods of sharing knowledge to allow for cooperation between research groups. This type of cooperation is necessary to ease the building of knowledge, evolution of experimental artifacts, and drawing conclusions across all studies.

Basili et al. provided one of the first guidelines for experimental software engineering replications, focusing on establishing reporting guidelines. The guidelines categorize the experimental process into four steps: Definition; Planning, Operations, and Interpretation [7]. Lott and Rombach also expanded on this four-phase structure by providing greater detail and direction [16]. Kitchenham et al. provide a more general and abstract set of guidelines to encourage critical assessment of existing studies. They cover a broader spectrum of software engineering studies, from observational studies to controlled experiments [12]. Wohlin et al. and Juristo and Moreno provide in depth guidelines for performing and reporting controlled experiments [10,29]. Brooks et al., provide guidelines to improve experimental 'recipes', the use of alternative data analysis techniques, and packaging experiments for replication [8].

Although guidelines such as these are an essential component in promoting good practices, they are not sufficient for ensuring reproducibility [19]. Some experimenters provide 'replication packages' or 'lab packages' to increase reproducibility of their work [4, 11] and the construction of families of experiments [5]. These packages include the experimental design and artifacts (forms, documents, and code) necessary to run an experiment.

The problem with most lab packages is that much information that needs to be made explicit remains tacit. Even when both the original experimenters and the replicators are experienced experimentalists, there are still many potential sources of variation and implicit assumptions made about the experimental context. Acquiring the necessary information about an experiment to guard against unintended sources of variation between replications is quite difficult, due in part to the tacit knowledge problem. Tacit knowledge is important information that is known only by the experimenter and is difficult to make explicit (that is, written down), for a variety of reasons [24,26].

## 3. The Perspective Based Reading Experiments

In this section, we illustrate our experience with a family of replicated studies that compare the use of a particular inspection technique, Perspective-Based Reading (PBR) to a more standard approach.

Typical techniques used by participants to find defects during an inspection include ad-hoc defect detection (where the detection process is unspecified and driven largely by the interests and experience of the inspector) and checklist based reading (in which each inspector focuses on a list of the quality aspects or defect types). PBR, in contrast, provides a more procedural approach. A PBR inspector is assumes the perspective of one of the stakeholders of the requirements document and creates an abstraction of the requirements relevant to that stakeholder. During the abstraction process, the inspector considers a set of questions, based on the important defect types, to help them locate defects. The set of perspectives used were a software designer (D), a tester (T), and an end-user (U). Each perspective helps the inspector ensure that the requirements document contains the appropriate information for the stakeholder to do his or her job. For example, the Tester perspective helps the inspector to uncover defects that would make the final product difficult to test [23].

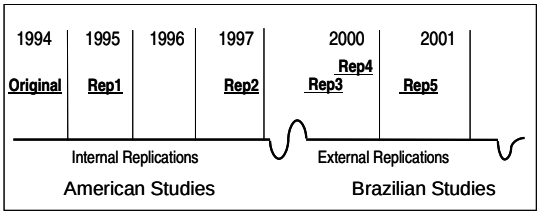


**Figure 1. Studies used to illustrate the FIRE**

The remainder of this section discusses the replications of this study. The first studies occurred in the American research group, where three studies were executed (the original study and two intra-group replications). Next, the Brazilian research group executed three external replications. As the replications are discussed, any evolution of the study (e.g. experimental design, subjects, or types of analyses) is highlighted. In addition, specific results are discussed that illustrate the growth knowledge through the replication. Throughout the rest of the paper the studies are numbered in the order in which they occurred (Original, followed by Rep1-Rep5). The location of each study is indicated with "US" for American studies and "BR" for Brazilian studies. Figure 3 presents an overview of the studies discussed in this paper.

### 3.1. The Internal Studies

Inspired by the successful application of code reading for software defect detection [2], researchers at the University of Maryland began investigating techniques for defect detection in requirements documents. At this point no existing experiments or lab packages were available, so knowledge was abstracted from a review of the literature on inspections and other related studies. Although planning the original experiment is not the same as planning a replication, no experiment starts completely from scratch – there is always explicit knowledge from general literature and other sources.

**3.1.1. The original study.** The **study goal** was to compare the effectiveness (number of defects found per subject) of PBR to the effectiveness of another approach termed the comparison technique. In this case, the comparison technique was the standard inspection method used by NASA software developers, an informal approach.

The **experimental design** is shown in Table 1. Eighteen subjects were obtained through publishing a call for volunteers at NASA's Goddard Space Flight Center. (Volunteers were offered free training in the inspection techniques as part of their participation in

**Table 1. Experimental design**

<table>
<tr><td></td><td colspan="3">Group 1 – 9 Subjects</td><td colspan="3">Group 2 – 9 Subjects</td><td></td></tr>
<tr><td rowspan="3">First Day</td><td colspan="3">Training (ABC video)</td><td colspan="3">Training (ABC video)</td><td rowspan="3">Usual Technique</td></tr>
<tr><td colspan="3">NASA A</td><td colspan="3">NASA B</td></tr>
<tr><td colspan="3">ATM</td><td colspan="3">PG</td></tr>
<tr><td rowspan="5">Second Day</td><td>Designer<br>3 Subjects</td><td>Tester<br>3 Subjects</td><td>User<br>3 Subjects</td><td>Designer<br>3 Subjects</td><td>Tester<br>3 Subjects</td><td>User<br>3 Subjects</td><td rowspan="5">PBR Technique</td></tr>
<tr><td colspan="6">Training on PBR technique</td></tr>
<tr><td colspan="3">Training (ABC video)</td><td colspan="3">Training (ABC video)</td></tr>
<tr><td colspan="3">PG</td><td colspan="3">ATM</td></tr>
<tr><td colspan="3">NASA B</td><td colspan="3">NASA A</td></tr>
</table>

the study.) On the first day, subjects performed an inspection using the NASA approach. On the second day, each subject was trained on one of the three PBR perspectives (6 subjects per perspective) and then performed an inspection using that perspective.

Four requirements documents were used in this experiment, two from generic domains: an Automated Teller Machine (ATM) and a Parking Garage (PG); and two from a NASA-specific domain: NASA A and NASA B. The subjects used one generic and one NASA document during inspection 1 and switched documents for inspection 2. To prevent the order of the documents from influencing the results, half of the subjects used NASA-A and ATM first followed by NASA-B and PG and the other half reversed the order. The study was designed so that the core of the study, the generic documents, could be replicated in a non-NASA environment. Replicating the study without the NASA documents was certainly a different design. But, without requisite domain knowledge, non-NASA inspectors would not be able to understand the NASA-specific documents. Therefore, in the replications reported in this paper, we focus only on the generic documents.

During the **execution of the study,** two sets of metrics were collected. First the subjects were given a survey to collect their background knowledge and experience. The second set of metrics measure the defects found by subjects. The average effectiveness of inspectors using PBR was compared to the average effectiveness of inspectors using the comparison technique (the standard NASA method). Also, random groups of either three *ad-hoc* subjects or three PBR subjects (one from each perspective) were created and the average effectiveness of these teams was compared. A short post-study questionnaire was administered to collect qualitative feedback about usefulness and difficulties encountered in using the techniques.

Because this was the first study, **analysis** covered the results from this study only and did not involve **integration** of the new results into an existing body of knowledge. A first attempt at **packaging the experiment** was made. At this point, the researchers did not fully understand what information needed to be in a laboratory package. This initial package did little more than document what had occurred. The researchers had not yet realized the difficulties with tacit knowledge, so little effort was spent in externalization of tacit knowledge.

The **results of this study** showed promising indications about the effectiveness of PBR: At both the individual and team level, inspectors using PBR tended to detect 20% more defects than inspectors using the comparison technique. However, due to the small number of subjects the difference was only statistically significant at the team level.

**3.1.2. The first American replication (Rep_1).** This replication had the same **goal** as the original experiment. This study was used to gather more data in the hopes of being able to make a more definitive statement about PBR's relative effectiveness. Based on the experiences from the original experiment, some modifications were made in **designing the experiment and adapting the artifacts:**

1) *Update of experimental artifacts*. Small changes were made to the experimental artifacts (the requirements documents) to remove some minor sources of confusion that did not impact the experimental goals.
2) *Update of design*: The experimental venue was moved from NASA's Goddard Space Flight Center to the University of Maryland, to avoid

work-related distractions that might affect the results of the inspection.

3) *Modifications to data analysis*: The master list of defects was refined to include defects that were present in the documents in addition to those that the experimenters had seeded. Moreover, the description of some defects was clarified to aid in the analysis phase. After these changes, the data from the Original Study were re-analyzed for consistency.

**Specific results.** The results showed a statistically significant improvement of about 30% at both the individual and team level when using PBR. The subjects also provided qualitative data about their experiences with PBR. The most experienced developers, in particular, seemed to have difficulty giving up their usual practices to apply a new technique under a time limit. The distribution of experience among subjects in this study was not sufficient to test this hypothesis, but it was flagged for additional study.

**Integrated results.** Since the direction and magnitude of the effects observed were consistent with those of the Original Study, the results of Rep_1 provided additional confidence in the effectiveness of PBR.

A major contribution of Rep_1 was the **lab package.** Between the original study and this replication, one of the original experimenters was replaced by a new researcher. Realizing how much effort was required for understanding the Original Study, this researcher developed the lab package to minimize this effort in the future. A complete description of the results of the Original Study and Rep_1 have been published in [3].

**3.1.3. The second American replication (Rep_2).** The second replication was the basis for a Ph.D. dissertation [22]. Its **goal** was to extend the results of the original study, and investigate a new question that arose in Rep_1. Specifically, did the relative effectiveness of the techniques vary for subjects with different levels of experience? In order to address this new question and better understand the extent of the applicability of the results, the subject population was changed. Rather than consisting entirely of NASA software engineers, subjects were drawn from students in a graduate-level software engineering course at the University of Maryland. Subjects thus ranged in experience from industry professionals returning for advanced degrees to researchers with no software development experience outside the classroom. A second change was that instead of using the standard NASA approach as the *comparison method*, the subjects used an ad hoc approach. The remainder of the **design of the experiment** remained unchanged. The artifacts and defect lists were again slightly updated based on the results from the previous studies.

The **analysis** was conducted largely the same as in the previous studies, except that a new independent variable (subject experience in their PBR role) was tested for correlation with effectiveness. This variable was measured on a three level ordinal scale (no industrial experience, industrial experience on one or two projects, and industrial experience on more than two industrial projects).

**Specific results**. The results showed that only medium experience subjects benefited from using PBR. The experimental team hypothesized that the cause may be that very inexperienced subjects did not have enough background in the given perspective to apply it effectively, while highly experienced subjects had likely developed their own procedures and were more comfortable and more effective using them.

**Integrated results.** Comparison of the results of Rep_2 to those from the previous studies showed some consistency. For example, the improvement by the subjects with a small amount of industrial experience, was comparable to that seen in past studies, lending additional support to the observation that PBR is useful for this type of inspector.

## 3.2. The External Studies

In the late nineties, supported by NSF and CNPq, Brazilian and American researchers established a collaborative research effort to studies effort software defect detection techniques. The effort was named the Reader´s Project. The first step of the project was to **plan studies and coordinate initiatives**. It was established that the overall goal of this project was to replicate experiments in different environments to gain a better understanding of their variations. A major focus was to investigate cultural issues that arise during replications in different environments. As part of this coordination, external replications of the PBR Experiment were conducted in São Paulo, Brazil.

In order to do that, the replicating researchers (the Brazilians) needed to **understand the experiment and lab package**. Obtaining the laboratory package and the associated artifacts was a key issue for these replications. Even though a lab package had been produced during the earlier studies, assembling a complete and consistent set of materials was not easy. Between 1997 and 2000, the lab package had been used by other external replications. It contained

artifacts that had evolved without proper version control and configuration, making the identification of compatible and consistent artifacts complicated.

By interacting with original experimenters the replicating researchers were able to obtain a *consistent* lab package, but later experience showed it was not entirely *complete* from the replicator´s viewpoint.

**3.2.1. The Brazilian pilot study (Rep_3).** The first step for the Brazilian researchers was to **set experimental goals**. The goals of Rep_3 were foremost to replicate the Original Study plus examine the effect of cultural differences. A change was made to the experimental design - the comparison inspection technique was a *checklist* instead of *ad hoc* or the informal *NASA method*. An *ad-hoc* inspection relies heavily on the background knowledge of the inspectors; therefore comparing the use of *ad-hoc* by inexperienced subjects to PBR would unfairly benefit PBR. So, a standard industry practice, a checklist, was used to provide a more fair comparison. In addition to those experimental goals, this pilot study had the goal of allowing the BR team to master the experimental procedure. Specifically, the BR team wanted to ensure experimental process conformity with the Original Study, including the activities, timing, and artifacts. Additionally, a minor modification was made to the original **study design** to collect a new metric: defect occurrences, which measures the degree of overlap among defects found by different subjects.

As the original experimenters were not present for the replications, this first study had to address experimental issues that were not explicit in the experimental package. **Adaptation of artifacts** for this pilot study focused on the BR replicators capturing and externalizing (i.e. documenting) the tacit knowledge about the experimental procedures, their timing, and their input and output artifacts. The six subjects were selected from Ph.D. students working in the software engineering area, with 3 using the checklist and 3 using PBR (1 per perspective).

Because Rep_3 was a pilot study, we did not report the specific results comparing PBR to the checklist. Rather the main result of the study was that the BR team was now confident enough in their understanding of the study to proceed with a full replication, Rep_4.

**3.2.2. The First Brazilian Replication (Rep_4).** The first complete external replication was conducted at the University of São Paulo at São Carlos (USP). The **experimental goals** and **experimental design** were the same as for Rep_3. Two small additions were made to the data collection forms. First, the background survey was augmented to characterize English experience because the subjects spoke Portuguese as their native language. Second, new questions were added to the post-study feedback questionnaire to gauge the subjects' understanding of the PG and ATM domains and their process conformance. In addition, a live feedback session was held at the conclusion of the study, where researchers presented the subjects with an overview of the experimental results and the defects found in the artifacts. The subjects were then asked to discuss whether they agreed with the defects identified by the researcher and describe how well they followed the process.

The **subjects** were 18 volunteer undergraduate students (i.e. no course credit was given for participating in the study) from the Software Engineering course at USP. Subjects were randomly distributed among the experimental groups ensuring that expertise in software development and English reading ability were balanced.

**Specific results.** The results from the **data analysis** show that for the ATM, the subjects using the PBR technique found a higher percentage of the defects than the subjects using Checklist, while the opposite was true for the PG. In attempting to understand why the results differed for the two artifacts, efficiency (number of defects found/hour) and defect occurrences were examined. The subjects using PBR were more efficient for both documents and found more defect occurrences for the ATM only. Regardless of which metric was used, there was little difference between the techniques for the PG.

**Integrated results.** The results of Rep_4 partially supported the results of the American studies. PBR performed better than the comparison technique on the ATM document, a result consistent with previous PBR experiments [3]. Conversely, it performed worse than the comparison technique on the PG document (for the effectiveness and occurrences metrics), which conflicts both with the ATM results in this replication and the PG results in previous studies. Individual efficiency using PBR was also better for both documents, but was not measured in the original study.

There are two potential explanations for the inconsistency between the results from the Original Study and these results for the PG document: 1) the change of the comparison technique from the *NASA/ad-hoc* to *checklist*, 2) the native language of the subjects was different than the language of the experimental materials.

The last step in Rep_4 was to **evolve the experimental package**. One issue that arose during the data analysis process was the difficulty of determining whether reported defects were true defects or false positives. The BR team determined that a list of frequently reported false positives should be part of the lab package to ease analysis in future studies. Second, the list of true defects was modified to include new defects found during Rep_4. These modifications were made through interaction with the original experimenters to ensure a shared understanding.

After Rep_4, a second BR replication, Rep_5, was planned. Moving from the Rep_4 to Rep_5 was considerably simpler than moving from the original lab packages to Rep_4 because the experiment was already understood and adapted by the BR team.

**3.2.3. The Second Brazilian Replication (Rep_5).** The second BR replication occurred at the Federal University of São Carlos. To maintain uniformity, no changes were made to the **experimental goals** from Rep_4. While the goals of Rep_5 did not change, the **experimental design** was slightly modified, based on the feedback from Rep_4: the checklist training and the checklist application were done on subsequent days, a week later the PBR training and PBR application were done on subsequent days. The total training time in each replication was the same; only it was spread out over a longer time period. These minor procedural changes are a possible source of variation in the results, but, in both studies, each technique was still applied within one day of its training.

Another difference between Rep_4 and Rep_5 was the **subjects**. Rep_5 used 18 undergraduate students from a Software Engineering course at the Federal University of São Carlos with slightly different motivation than those who participated in Rep_4. Only 1/3 of the subjects in Rep_5 were volunteers, the other 2/3 were given course credit for participation in the study. Subjects were distributed among the experimental groups to balance expertise in software development and English reading ability.

**Specific results.** The results from the **data analysis** show that the subjects using the PBR technique found a higher percentage of defects and were more efficient than the subjects using the Checklist for both the ATM and PG (neither result was statistically significant). For the occurrences metric, subjects using PBR performed better on both documents (no statistical tests were run). A complete description of Rep_4 and Rep_5 and their data analysis has been published in [17].

**Integrated results.** Because the subject populations and the experimental procedures for Rep_4 and Rep_5 were similar, the data from the two studies were pooled and reanalyzed. This analysis showed that the subjects using PBR were both more effective and efficient on both ATM and PG (the ATM effectiveness and efficiency were significant). These results are consistent with the results from the American studies.

Contrary to the subject feedback from Rep_4, who suggested the experiment should not run on consecutive days, subjects in Rep_5 stated that they would rather have the experiment run on consecutive days. This discrepancy shows that subjects sometimes give conflicting feedback.

The last step in Rep_5 was to **evolve the experimental package**, based on Rep_5:

1) Update of experimental artifacts. The questions on the post-study feedback questionnaire were organized topically.
2) Modification to training. This study was spread out over a one-week period. However, the subjects stated they would prefer the study to run in consecutive days. In the lab package, this issue remains open for replicators to decide.
3) Modification to data analysis. The need for a standard data spreadsheet format was identified to minimize difficulties of integrating results with those from previous studies.

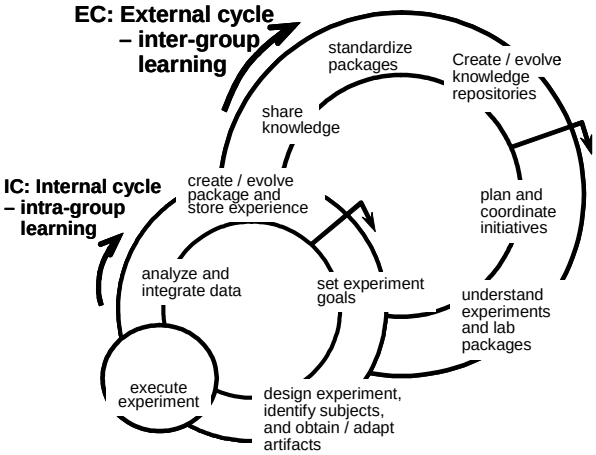


**Figure 2. The FIRE**

## 4. A Framework for Improving the Replication of Experiments (FIRE)

Based on our experiences on the Reader´s Project, we proposed a *Framework for Improving the Replication of Experiments (FIRE)*. The FIRE is inspired by the

Quality Improvement Paradigm (QIP) [1]. As seen in Figure 2, the FIRE contains two cycles. In the internal (experiment execution and intra-group learning) cycle (IC), experimenters focus on independently and successfully **planning**, **executing**, **learning** and **packaging** the experiment within their own context. In the external (inter-group learning) cycle (EC), experimenters are concerned with collaborative package standardization, experimental knowledge evolution, and knowledge sharing.

The FIRE recognizes that an experiment demands precise execution and any deviations from planning are closely monitored; an experiment is usually aborted if it deviates too much from planning. So, the internal cycle of FIRE focuses on the implementation of a single experiment. The FIRE External Cycle (inter-group learning) involves learning across groups. However, the emphasis on the FIRE external cycle is on inter-group learning, where groups are located in different organizations, geographically distributed and culturally diverse. Sharing experimental knowledge among research groups is the very nature of software engineering experimentation as the basis for experimental repeatability, process uniformity, and knowledge base growth.

The goal of the inter-group learning cycle is to build a broad body of explicit knowledge from carefully planned replications executed by different research groups. Based on the Reader´s experience, the key activities of this cycle are: (1) **plan and coordinate experimental activities** among the research groups to increase experimental knowledge; (2) **understand lab packages and results** from other research groups (the effort required depends largely on the amount of coordination among the groups); **execute the intra-group learning** cycle (independently conduct replications and evolve the lab package by tailoring the experimental design and artifacts); (3) **share and consolidate new knowledge** with other groups; (4) **standardize packages**; (5) **evolve the body of knowledge**. The input to the IC is the explicit results and procedures from various replications. The output of the IC is evolved knowledge that can be used as the basis for a new replication.

While the inter- and intra-group cycles are well understood in isolation, difficulties arise at the interfaces between these cycles. Experimenters often cannot anticipate all the design details and rationale that will be most relevant to replicators, so making the right information explicit in the inter-group cycle is difficult. Similarly, it is difficult for replicators themselves to determine the most relevant context variables for to allow for meaningful comparison of results.

As an example of the difficulties at the cycle interfaces, a researcher who spent a sabbatical with an experimental group associated with the University of Maryland replicated an inspection study in his native country. In spite of his exposure to the research group, and the availability of a laboratory package, this experimenter still faced problems when adopting the experiment to subjects with different cultures, languages, and backgrounds. One example of this was that a time limit was used for the inspection which may have been appropriate for the original subjects who were working in their native tongue, but which seemed too constrained given that the subjects in the replication were less familiar with the problem domain and were working in a language with which they felt less comfortable. An unanticipated result of this seemingly minor change was that the subjects found themselves unable to complete much of the task in the time allowed. The subjects reported feeling frustrated and de-motivated with the procedures they were using and quite possibly performed them less effectively as a result. In short, although the researcher who conducted the replication had tacit knowledge that most replicators would not possess, the results from his replication were not comparable to those from the original study: The results in the two contexts were quite different, due primarily to the way the experiment was run, and not due to an intrinsic property of the inspection techniques themselves.

For this reason the key step in the FIRE cycle interfaces is **knowledge sharing**. In the Reader´s Project, the groups involved listed the following initiatives as key factor for the project success:

1) Frequent interaction among groups through e-mail and phone calls;
2) Execution of pilot studies, and;
3) Regular presential workshops to synchronize initiatives and to discuss the experiments and its artifacts. Eight workshops were held during the four years of the project. In the case of the PBR experiments this led to:
   a. A more precise description of the experimental process;
   b. Modification of the background questionnaire to include language issues;
   c. New defects detected; and
   d. Identification of difficulties in handling the false positives defects in data analysis.

The difficulties with combining and analyzing data from different experiments were also discussed. Much attention was given to the subjective measures collected in the background and post-study feedback questionnaires, resulting in two new research topics for future studies: (1) There are several potential variables in the subject profile (e.g. experience with technique, language expertise) that need to be measured more accurately; and, (2) The elapsed time that for each defect report may provide interesting insight into the use of the different inspection approaches.

After these issues were discussed the process of **package standardization** occurred by evolving the original lab package to include the modifications introduced during the BR replications that were agreed upon by both teams: (1) An updated defect list; (2) A false positive list; (3) The identification of a process for updating the false positive and defect lists; (4) An evolved description of the experimental process was adopted; (5) The pilot study was included in the standard experimental design; (6) The background questionnaire was modified to produce a standard questionnaire for all experiments conducted in the Readers' project; and (7) A defect report time field was added to the defect report form.

The last step of FIRE is to assemble all the information from the original experiment and the replications into a body of knowledge. In the case of the Readers, the US researchers had already identified the need for a common repository of experimental knowledge [6], and begun working to aggregate the knowledge from the PBR experiments into this repository.

## 5. Conclusion

This article presents our experiences in executing experimental replications in software engineering. It argues that knowledge sharing is a key issue in executing replication and building a body of knowledge in software engineering. If replications focus only on acquiring a lab package and following the steps in the intra-group learning cycle, they run the risk of becoming isolated studies that are hard to integrate into a larger body of knowledge. Our experiences indicate that replicators can profit from: (1) Coordinate initiatives; (2) Master the explicit and tacit knowledge associated with the lab packages; (3) Set the goals and scope of their replications; (4) Carefully and independently execute the replications; (5) Analyze the data and compare with previous data; (6) Share findings with the original experimenters; and (7) Work with the original experimenters to evolve and standardize related lab packages.

We are not arguing against the need for independent replications, as they are at the very heart of science and useful to confirm and expand results. Independent replicators are less susceptible to bias from the original experimenters because they bring a fresh perspective, which may lead to new results. Conversely, coordinated replications ease knowledge building, the artifact evolution, and the cross-study analysis. The seven steps of the FIRE are geared toward these three issues. It assumes that researchers are collaborating closely using a variety of communication mechanisms, e.g., small, intense workshops. Such mechanisms are necessary because the socialization of knowledge at the inter-group level is more difficult than at the intra-group level. When a researcher becomes disconnected and unable to coordinate research efforts, he, and the community as a whole, partially lose the ability to harmonize and consolidate results.

Also, running experiments in isolation from a community is very difficult. There is a recognized need for collaboration, local support, and a culture of experimentation. This realization motivated the creation of the International Software Engineering Research Network (ISERN)[1]. Its founding members had been early collaborators that felt the need for a support community once they left the group.

Once again we stress that close collaboration does not imply that replicators should not be critical of the original experiment, evolve it, or improve and produce new experimental artifacts. In fact, there is an advantage to multiple groups reviewing an experimental design. The FIRE is spit into two cycles specifically to allow replicators to act independently during the execution of the replication to reduce bias from contact with the original experimenters.

## 6. References

[1] Basili, V. "Quantitative Evaluation of Software Engineering Methodology". In Proceedings of 1st Pan Pacific Computer Conference. Melbourne, Australia. 1985 p. 379-398

[2] Basili, V. and Selby, R., Comparing the Effectiveness of Software Testing strategies. IEEE TSE, 1987. 13(12): p. 1278-1296.

[3] Basili, V., Green, S., Laitenberger, O., Shull, F., Sorumgaard, S., and Zelkowitz, M., The Empirical Investigation of Perspective Based Reading. Empirical

[1] http://www.iese.fhg.de/ISERN

Software Engineering - An International Journal, 1996. 1(2): p. 133-164.
[4] Basili, V., Evolving and Packaging Reading Technologies. Journal of Systems and Software, 1997. 38(1): p. 3-12.
[5] Basili, V., Shull, F., and Lanubile, F., Building Knowledge through Families of Experiments. IEEE TSE, 1999. 25(4): p. 456-473.
[6] Basili, V., Tesoriero, R., Costa, P., Lindvall, M., Rus, I., Shull, F., and Zelkowitz, M. "Building and Experience Base for Software Engineering: A report on the first CeBASE eWorkshop". In Proceedings of PROFES (Product Focused Software Process Improvement). 2001 p. 110-125
[7] Basili, V.R., Selby, R.W., and Hutchens, D.H., Experimentation in software engineering. IEEE TSE, 1986. SE-12(7): p. 733-743.
[8] Brooks, A., Roper, M., Wood, M., Daly, J., and Miller, J. Replication of Software Engineering Experiments. Empirical Foundations of Computer Science Technical Report, EfoCS-51-2003. Department of Computer and Information Sciences, University of Strathclyde: 2003.
[9] Brooks, R., Studying Programmer Behaviour Experimentally: The Problems of Proper Methodology. Communications of the ACM, 1980. 23(4): p. 207-213.
[10] Juristo, N. and Moreno, A., Basics of Software Engineering Experimentation. 2001: Kluwer Academic Press.
[11] Kamsties, E. and Lott, C. An Empirical Evaluation of Three Defect-Detection Techniques. International Software Engineering (ISERN) Technical Reports, ISERN-95-02. 1995.
[12] Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., and Rosenberg, J., Preliminary guidelines for empirical research in software engineering. IEEE TSE, 2002. 28(8): p. 721-734.
[13] Laitenberger, O. and DeBaud, J.-M., Perspective-Based Reading of Code Documents at Robert Bosch GmbH. Information and Software Technology, 1997. 39(11): p. 781-791.
[14] Laitenberger, O., Atkinson, C., Schlich, M., and El Emam, K., An Experimental Comparison of Reading Techniques for Defect Detection in UML Design Documents. Journal of Systems and Software, 2000. 53(2): p. 183-204.
[15] Laitenberger, O., El Emam, K., and Harbich, T.G., An internally replicated quasi-experimental comparison of checklist and perspective based reading of code documents. IEEE TSE, 2001. 27(5): p. 387-421.
[16] Lott, C. and Rombach, D., Repeatable Software Engineering Experiments for Comparing Defect-detection Techniques. Empirical Software Engineering, 1997. 1(3): p. 241-277.
[17] Maldonado, J., Carver, J., Shull, F., Fabbri, S., Doria, E., Martimiano, L., Mendonca, M., and Basili, V., Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness. Empirical Software Engineering, 2006. 11(1): p. 119-142.
[18] Miller, J., Applying meta-analytical procedures to software engineering experiments. Journal of Systems and Software, 2000. 54(1): p. 29-39.
[19] Miller, J., Replicating Software Engineering Experiments: A Poisoned Chalice or the Holy Grail. Information and Software Technology, 2005. 47(4): p. 233-244.
[20] Porter, A.A., Votta, L.G., Jr., and Basili, V.R., Comparing detection methods for software requirements inspections: a replicated experiment. IEEE TSE, 1995. 21(6): p. 563-575.
[21] Regnell, B., Runeson, P., and Thelin, T., Are the perspectives really different? Further experimentation on scenario-based reading of requirements. Empirical Software Engineering, 2000. 5(4): p. 331-356.
[22] Shull, F., Developing Techniques for Using Software Documents: A Series of Empirical Studies, Ph.D. Thesis, Department of Computer Science, University of Maryland, College Park, 1998
[23] Shull, F., Rus, I., and Basili, V., Perspective-Based Reading: Techniques for Improving Requirements Inspections. IEEE Computer, 2000. 33(7): p. 73-79.
[24] Shull, F., Basili, V., Carver, J., Maldonado, J., Travassos, G., Mendonca, M., and Fabbri, S. "Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem". In Proceedings of ISESE'02. Nara, Japan. Oct. 3-4, 2002, 2002 p. 7-16
[25] Shull, F., Carver, J., Travassos, G., Maldonado, J., Conradi, R., and Basili, V., Replicated Studies: Building a Body of Knowledge about Software Reading Techniques, in Lecture Notes on Empirical Software Engineering, N. Juristo and A. Moreno, Editors. 2003, World Scientific.
[26] Shull, F., Mendonca, M., Basili, V., Carver, J., Maldonado, J., Fabbri, S., Travassos, G., and Ferreira, M., Knowledge-sharing Issues in Experimental Software Engineering. Empirical Software Engineering - An International Journal, 2004. 9(1): p. 111-137.
[27] Sjoeberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.K., and Rekdal, A.C., A survey of controlled experiments in software engineering. IEEE TSE, 2005. 31(9): p. 733-753.
[28] Tichy, W.F., Should computer scientists experiment more? IEEE Computer, 1998. 31(5): p. 32-40.
[29] Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., and Wesslen, A., Experimentation in Software Engineering: An Introduction. 2000: Kluwer Academic Publishers.
[30] Zelkowitz, M.V. and Wallace, D.R., Experimental models for validating technology. IEEE Computer, 1998. 31(5): p. 23-31.
[31] Zendler, A., A Preliminary Software Engineering Theory as Investigated by Published Experiments. Empirical Software Engineering, 2001. 4(1): p. 43-70.
[32] Zhang, Z., Basili, V., and Shneiderman, B., Perspective-based Usability Inspection: An Empirical Validation of Efficacy. Empirical Software Engineering - An International Journal, 1999. 4(1): p. 43-70.