An Empirical Methodology for Introducing Software Processes

Forrest Shull
Fraunhofer Center - Maryland
University of Maryland
4321 Hartwick Road, Suite 500
College Park MD 20742
301-403-2705
fshull@fc-md.umd.edu

Jeffrey Carver
Experimental Software Engineering Group
Department of Computer Science
A.V. Williams Bldg.
University of Maryland, College Park
College Park MD 20742
301-405-2721
carver@cs.umd.edu

Guilherme H. Travassos
Computer Science Department
COPPE/UFRJ
C.P. 68511 - Ilha do Fundão
Rio de Janeiro – RJ – 21945-180, Brazil
55 21 562-8712
ght@cos.ufrj.br

ABSTRACT

There is a growing interest in empirical study in software engineering, both for validating mature technologies and for guiding improvements of less-mature technologies. This paper introduces an empirical methodology, based on experiences garnered over more than two decades of work by the Empirical Software Engineering Group at the University of Maryland and related organizations, for taking a newly proposed improvement to development processes from the conceptual phase through transfer to industry. The methodology presents a series of questions that should be addressed, as well as the types of studies that best address those questions. The methodology is illustrated by a specific research program on inspection processes for Object-Oriented designs. Specific examples of the studies that were performed and how the methodology impacted the development of the inspection process are also described.

Categories and Subject Descriptors

K.6.3 [**Management of Computing and Information Systems**]: Software Management –*Software process.*

General Terms

Measurement, Design, Experimentation, Verification.

Keywords

Empirical studies, OO design inspections, software process, experimental process, software quality

1 CHALLENGES OF SOFTWARE TECHNOLOGY INTRODUCTION

There is a growing interest in empirical study in software engineering, as evidenced by the growing number of publications incorporating empirical methods [23] and increasing investment in empirical research (e.g. NSF's CeBASE project [5]). Using empirical studies to study software development under realistic conditions can provide validation for mature technologies – assessing the effectiveness of proposed development tools and methods in various environments – and identification of the problems present in less mature technologies.

This paper introduces an empirical methodology, based on experiences garnered over more than two decades of work by the Empirical Software Engineering Group at the University of Maryland and related organizations, for taking a newly proposed development improvement from the conceptual phase through transfer to industry. This methodology represents what we have learned about balancing the needs of research and industry throughout the process. This methodology has been abstracted from lessons learned on multiple research programs, for example understanding Cleanroom techniques [3] and PBR techniques for requirements inspections [2].

Although we claim this methodology will be useful for any software development process (e.g. new improvements in compilers and automated test methods), for the purpose of a concrete discussion in this paper we restrict our discussion to software processes[1]. In later sections of this paper, we present examples of the methodology used to guide a specific research program on inspection processes for Object-Oriented (OO) designs.

Process definition is important

The argument for defining processes for specific software development tasks should be a familiar one. A well-defined process can be observed and measured, and thus improved. Processes can be used to capture the best practices for dealing with a given problem. The adoption of processes also allows for dissemination of effective work practices to occur more quickly than the building up of personal experience. An emphasis on process helps software development become more like engineering, with predictable time and effort constraints, and less like art. [13]

[1] By "process" we are referring to a set of procedural guidelines for accomplishing specific development tasks, such as inspections. The type of process that we discuss in this paper will fit inside of development structures such as the waterfall lifecycle or frameworks like the CMM.

Process definition requires iteration

No matter how good an idea is on its own merits, there are many other factors that influence its usefulness: budget and effort constraints, practical usefulness, etc. Many of these factors simply can not be assessed in a laboratory environment.

These factors are exactly why technology transfer is not a trivial task. They are also the reasons why studies of new software development processes, inserted into industrial processes, are high risk. If a new process does not demonstrate significant improvement in an industrial environment, it is often difficult to know which factor was the likely cause: whether the basic idea itself was faulty, the new process did not fit into the industrial process, or the process itself was correct but applied incorrectly. In contrast, an iterative approach, in which an effort is made to separate out these factors and test them one at a time, has a better chance of resulting in real understanding.

A second reason for adopting an iterative approach is the necessity of tailoring. Because each development environment is unique, there is no such thing as a *one size fits all* process [16]. An iterative approach ensures the fundamental issues are addressed before a process is fine-tuned to a particular environment.

An iterative approach has value because it allows the use of resources to get the most benefit out of a given study, and allows studies to build upon each other more effectively. The alternative is a one-shot or "big bang" approach, in which a new idea is developed into a full-blown process and immediately tested. This approach can allow a quick demonstration of a process' effectiveness, or it can demonstrate that the process is unfit but provide little indication for how to address the problems (or even what the problems may be).

An iterative methodology for process definition

The Quality Improvement Paradigm (QIP) [1] has been recommended as a model for continual, iterative process improvement. The QIP is a set of six steps (characterize, set goals, choose process, execute, analyze, package), based on the scientific method, that are executed in a repeated cycle. The QIP provides an improvement framework but does not describe how exactly to study the process on any given iteration. The rest of this paper addresses just this issue. On each cycle of the QIP, a goal needs to be chosen, largely based on the results of the previous cycle. Each time through, the process becomes better understood and the improvements can be more effective.

2 EMPIRICAL STUDIES

Many types of empirical studies exist that are useful in software engineering. However, our experience has been that certain types of studies and strategies for data collection are more relevant to process work than others, and even those can be better suited to specific stages of process evolution.

The first important distinction to make is between qualitative and quantitative data. *Quantitative data* (numerical) is useful for measuring a particular aspect of a process, such as "number of defects detected", while *qualitative data* (expressed in words) is useful for getting a much richer understanding. Both types of data are necessary to evolve processes. Quantifying the effects of a process supports decision-making, but much useful insight will also come, in qualitative form, from the subjects executing the process in ways that are not easily reducible to quantitative data. The empirical methodology introduced in the next section is concerned with getting the right kind of data necessary to support the important decisions at a given time. Data collection should always be focused on useful measures of effectiveness (which are usually quantitative) but should include measures of feasibility or fit to the environment (which may be qualitative) whenever possible.

There are two types of lessons learned from empirical studies: *global* lessons that affect the entire process, versus *specific* lessons that affect individual process steps. Individual studies tend to concentrate on one or the other, either validating the overall focus and direction of a process or fine-tuning the individual steps to increase effectiveness. An important feature of our methodology is that global issues are addressed early, and specific ones late in the process. This feature helps guarantee that resources are conserved by making increasingly smaller refinements over time.

The next distinction is using either real professionals or using students in a classroom setting as subjects [8]. Students as subjects introduce certain validity concerns into any experiment since it is not clear which, if any, of their results hold for professional developers. Also, experiments on students have to be carefully designed for their pedagogical value. This means that because typically all the students in the class must be taught the new process, controlled experiments are not always possible. However, where possible, student subjects are valuable for debugging processes before introducing them in industry. Software professionals' time is so valuable that extensive measures are worthwhile to ensure that experiments in industry yield as much value as possible. Data collection should always include the past experience of the subjects applying the process. Our own experiences have shown this to be one of the most important factors influencing the effectiveness of a process. Subjects almost always react differently to a process based on their past experience.

Data collection of all types in empirical studies must address the question of process conformance. Empirical results are not of much use if the researcher cannot be sure of which process produced them! Some strategies for addressing this issue are found in [4].

And, data analysis has to be very sensitive to potential threats to validity. That is, although not all external factors can be controlled, a well-designed experiment will collect as much data as possible on the external factors in order to reason about their impact. Such threats cannot always be eliminated – but they do have to be enumerated and taken into account during analysis.

3 A METHODOLOGY FOR EMPIRICAL VALIDATION OF PROCESSES

In this section we provide a description of the methodology for the incremental evaluation of a new process. This methodology does not assume any particular origin for the new process, nor does it include the creation of the new process, which we can only emphasize must be based on observation of the real problems being faced by industry as well as the effectiveness of existing approaches in practice. This is not a trivial undertaking; many conferences, panels, and workshops have focused on improving

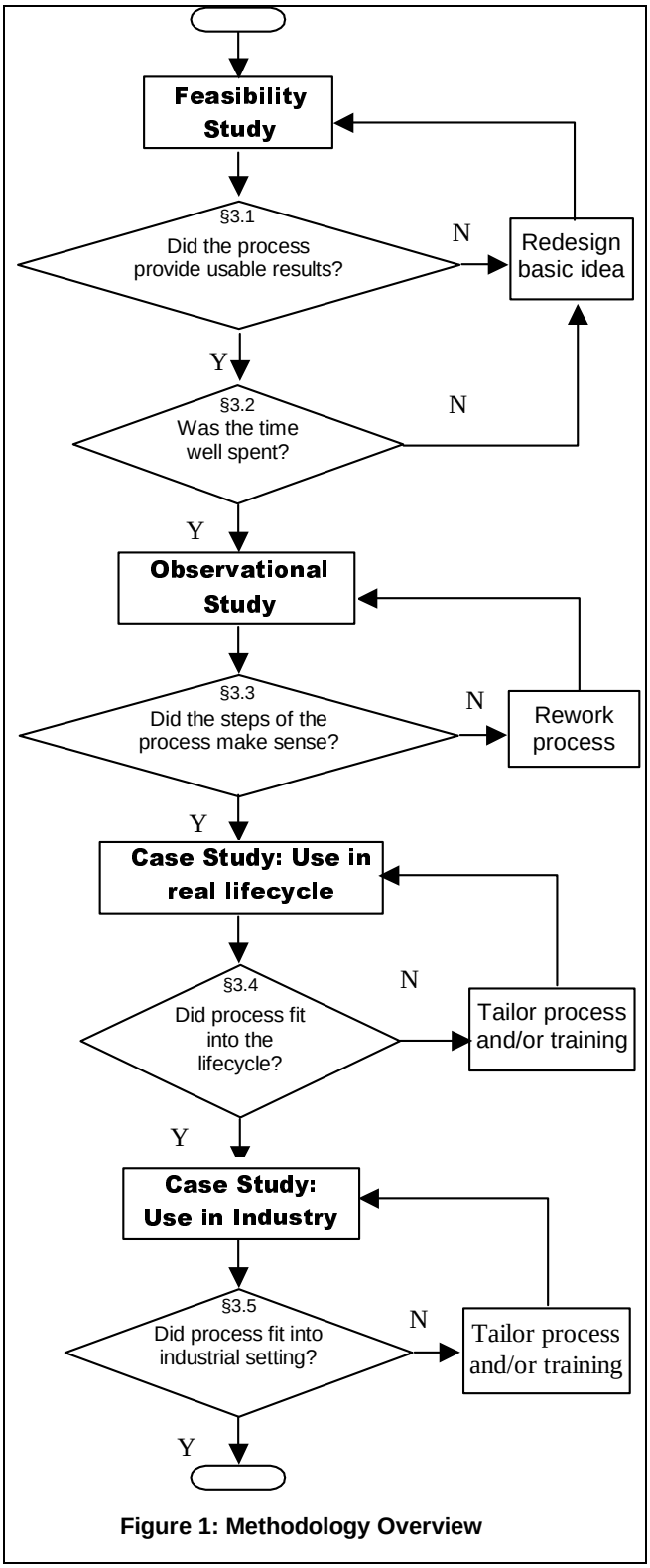

Figure 1: Methodology Overview

this interaction [7,18,22].

The flowchart in Figure 1 provides an overview of this methodology. The ordering of the questions in the flowchart forces one to examine the larger issues first and discover the larger problems early to avoid wasted effort later. Therefore, the questions that appear early in the methodology explore the basic, fundamental issues of the new process, which could prompt large changes. Later questions address issues that are more detailed and require smaller changes to fix. The remainder of this section addresses each question in the flowchart in more detail.

3.1 Did the process provide usable results?

This question evaluates if the new process fulfilled the overall goal for which it was created. At this level, researchers are evaluating if it is worthwhile to spend the resources required to continue through the methodology. In order to gather this type of information, researchers should use **feasibility studies** (sometimes referred to as "quasi-experimental designs" [6]) in which data is collected according to some experimental design, but full control over all possible variables is not achieved. Such studies attempt to test the effectiveness of a process but are not able to rule out all rival hypotheses that may still exist at the end of the study. For example, we may observe changes in subject effectiveness but cannot completely rule out the possibility that they were caused by something other than the new process. The goal here is to provide the researcher with enough information to justify continued work.

In order to provide the information, the effectiveness of the new process must be evaluated. Using qualitative data, rival hypotheses can be addressed. For example, the subjects may be asked if they think that their effectiveness improved because of the new process or because of some other reason. Or, multiple studies may be run such that no one study gives a definitive answer but each study attempts to rule out a different set of rival hypotheses. In either case, the objective is not to find a definitive answer but to build up a body of knowledge that addresses the plausibility of the process' effectiveness [4].

At this point in the methodology, we are concerned with generating rather than testing hypotheses about the new process and its usefulness. Therefore, classroom environments are well suited to feasibility studies. Although their results cannot be applied directly to industrial developers, running studies in the classroom allows new concepts to be tested before using them with expensive developers from industry. And, given that undergraduate and graduate students are bringing more and more industrial experience with them, one of the major threats to applicability of results is diminishing in importance [8].

3.2 Was the time well spent?

Once we have determined that the new process produces usable results, the next step is to determine whether or not the return on investment is reasonable, i.e. whether those results could have been achieved in a more cost-effective way. This type of information can be gathered based on questionnaires or interviews that piggyback on feasibility studies. Questionnaires and surveys are useful ways to collect both qualitative and quantitative data. These methods require a relatively small time investment from the experimenter, but sometimes cannot collect the information at the desired level of detail. While designing effective questionnaires does take some time, the fact that the same questions are sent out to all subjects makes it easier to aggregate the answers to give a quick snapshot of responses. Even with open-ended questions where subjects are allowed to write free-form answers, the

questions still provide a framework or initial categorization for the qualitative responses.

The questions can range from multiple choice (requiring the least subject time to answer, but providing less insight) to long answer (in which the subject can explain his or her reasoning in more depth, but answers are harder to compare across subjects). There should be enough choices for the multiple-choice questions to provide a reasonable degree of granularity, but not so many that the subjects cannot easily understand the distinctions between the choices. Having too many choices also runs the risk that few subjects will fall into any given category, complicating the task of abstracting patterns from the data. However, categories can be designed so that they can be grouped together during analysis; finding patterns across groups may be easier. Long answer questions give the subjects a chance to express their thoughts in more detail, but complicate the qualitative analysis (picking out patterns from free-form responses can be difficult). Such qualitative analysis is often the best way to gain insight into the use of the process.

Interviews are another method for collecting qualitative data. They are more time-consuming for experimenters, because they have to schedule and spend time with each subject to collect the data. However, the compensating benefits of interviews are that they allow for more freedom in the responses than questionnaires and surveys do. This freedom allows the interviewee to convey information in a way that makes sense to him or her. Interviews can also be dynamic because they allow the researcher to investigate topics he or she might not have even known were important prior to the interview. On the other hand, there are problems with accuracy because the subject cannot be anonymous and, consciously or not, the interviewee may want to "please" the researcher with his or her answers.

3.3 Did the steps of the process make sense?

Because at this point we have determined that the process produces useful results and also can be done with a reasonable amount of effort, we can begin to make more detailed modifications to the process. In order to do this, we begin by evaluating the steps in the new process to ensure that each one is effective and that the order in which they are executed makes sense. The most effective way that we have found to gather this type of information is through an **observational study**. We use the term "observational" to define a setting in which an experimental subject performs some task while being observed by an experimenter. The purpose of the observation is to collect data about how the particular task is accomplished. Observational techniques can be used to understand current work practices that can be incorporated into the new process. They are also useful for getting a fine-grained understanding of how a new process is applied. The observer is there to capture information about the circumstances in which the subject experiences problems or has trouble understanding the new process. The observer can also take note of the time consumed by each step of the process and whether or not the step was effective in achieving its goal.

An observational approach can be a bit more time-consuming for the experimenter and less relaxed for the subject than interviews or questionnaires. However, we have found that the observational approach delivers more accurate qualitative results than such *retrospective* methods. When retrospective methods are used subjects may find it difficult to reconstruct their own thought processes, or may (intentionally or accidentally) present their thought processes in a more structured and coherent way than actually occurred.

Data collection in an observational study can be split into two subtypes, observational and inquisitive. *Observational data* is collected while the process is being executed, but without interference from the researcher. For instance, subjects are told to think out loud as they execute a technique so that the researcher can gain insight into how the process is executed. For example, the researcher can record places that the subject becomes confused or does not know what to do next. Because the researcher should avoid interfering with the process, observational data collection is mostly passive [17].

Inquisitive data is collected at the completion of a process step, rather than during its execution. The researcher is required to be more assertive and to solicit responses to predefined questions rather than passively observe. For example, at the end of each step, the researcher could ask the subject for qualitative feedback as to whether that step was worthwhile or if the same results could have been better achieved in a different way. This is not information that the subject would normally think about while executing the process, yet it is invaluable to collect at this time, while it is still fresh in the mind of the subject.

3.4 Did the process fit into a particular lifecycle?

After the previous step, we have some indication that the process is effective. However, up to this point we have only used the process in isolation. In order for the process to really be useful, it has to be able to fit into a real development lifecycle. To find out this information, we can make use of a **case study**. Case studies examine a particular process in the context of a larger software lifecycle. We introduce case studies at this point in our methodology because they are not suitable vehicles for understanding a completely new process. They are expensive – subjects must be trained and must overcome the learning curve, and their time is potentially costly. Also, if an untested process is tried in the context of a lifecycle, it will be difficult when problems arise to differentiate between problems with the process itself versus problems with the interaction of the new process and the lifecycle process.

Case studies can incorporate different levels of rigor, ranging from more controlled studies (looking at a real lifecycle, but using controls such as a replicated or "baseline" project to study particular variables) to more realistic (done in industry with real time and budget pressures and professional developers, as well as many uncontrolled factors that can potentially influence events). More about case studies can be found in an overview paper by Kitchenham *et al* [9].

The goal of using a case study at this point in the development of the new technology is to do some fine-tuning or tailoring of the technology. Based on the previous steps in the methodology, we have already determined that the new process is promising and worth our effort to improve. At this point we begin to see how the new process interacts with other aspects of a real development lifecycle. This interaction can lead to issues that did not arise when evaluating the process in isolation. It is also important to remember that a process can be feasible and effective in some environments but not others. This step is a necessary one for

assessing compatibility with a particular development lifecycle – e.g. whether the right information is available for input, and whether the outputs are appropriate for later stages.

3.5 Did the process fit into an industrial setting?

Once we have tailored the new process to be usable within a real development lifecycle, the next step is to use the new process in an industrial setting. We again use a case study to investigate if the new process has any unforeseen negative interactions with the industrial setting.

We have placed this step last in our methodology because industrial developers are the most expensive of all the subjects that we have discussed. Therefore, we want to make sure that our new process is as good as possible before asking an industrial partner to invest their time and money. We should also remember as we begin to transfer this process to the organization that our goal is to minimize the interruption or disruption in the normal working environment.

For example, the researcher can spend time with the process owner of the organization discussing the new process and how it may or may not fit into the industrial setting. With the researcher being the expert on the process and the process owner being the expert on the industrial setting, some of the potential problems can be discovered before the expensive developers use the process. The more tailoring that can be done before the developers use the process, the more time and effort can be saved in this evaluation process.

4 APPLYING THE METHODOLOGY: READING TECHNIQUES FOR OO INSPECTIONS

We applied the methodology described in section 3 while building techniques that allow high level object-oriented designs to be inspected in order to detect defects. Previous research has shown that inspections can be improved by reading techniques, which assist inspectors to find defects while they individually review the document [11,14,24]. Because OO designs are quite different from structured designs, and increasing in popularity, new reading techniques tailored to the OO world are needed.

An OO design is a set of diagrams representing real world concepts in the problem domain, but built at a different time, using a different viewpoint and abstraction level than the requirements. When high-level design activities are finished, the diagrams can be inspected to verify whether they are consistent among themselves (*horizontal reading*) and if the requirements were correctly and completely captured (*vertical reading*) [19, 20]. We have developed a family of 7 reading techniques to compare various design diagrams, 4 horizontal and 3 vertical.

Ensuring the quality of the high-level design has benefits for software quality. First, by focusing the techniques on the high-level design, we are ensuring that developers understand the problem fully before trying to define the solution in the low-level design. Secondly, it is important to locate and remove as many defects as possible at the higher level, because they become more difficult and more expensive to fix if they are allowed to filter down into the low-level design, or even the code [10, 12].

The methodology from Section 3 was used to develop this new family of Object-Oriented Reading Techniques (OORTs) via a series of empirical studies at the University of Maryland beginning in 1998. The sequence of studies and evolution of goals illustrated in Figure 2, will be explained in sections 4.1-4.4.

4.1 Do OORTs provide usable results and is the time well spent?

Based on lessons learned from studying requirements inspections and different types of OO design defects, an initial set of reading techniques was created. Using the methodology described in Section 3, we first performed initial validation of the new techniques by using a feasibility study [19] in the Fall of 1998. This study addressed the questions asked in Sections 3.1 and 3.2 with feedback on form and content as a secondary goal.

Subjects: The subjects were students from a senior-level undergraduate software engineering course. Of the 44 students in the class, 32% also had previous industry experience in software design.

Materials: The initial version of the reading techniques was evaluated. All review teams applied them to the design of a small system (11 classes in high-level design).

Procedure: After students read the requirements for a system, the design was distributed to the class and subjects were asked to inspect it. There was no control group. Therefore, we could not compare the OORTs' effectiveness to that of another OO inspection method. There were two reasons for this decision. The first was that we were aware of no other published methods for reading OO designs. Secondly, it was a classroom environment. Each team applied all seven of the reading techniques, divided up among the members. After performing their individual reviews, the team members met to compile their individual defect lists into

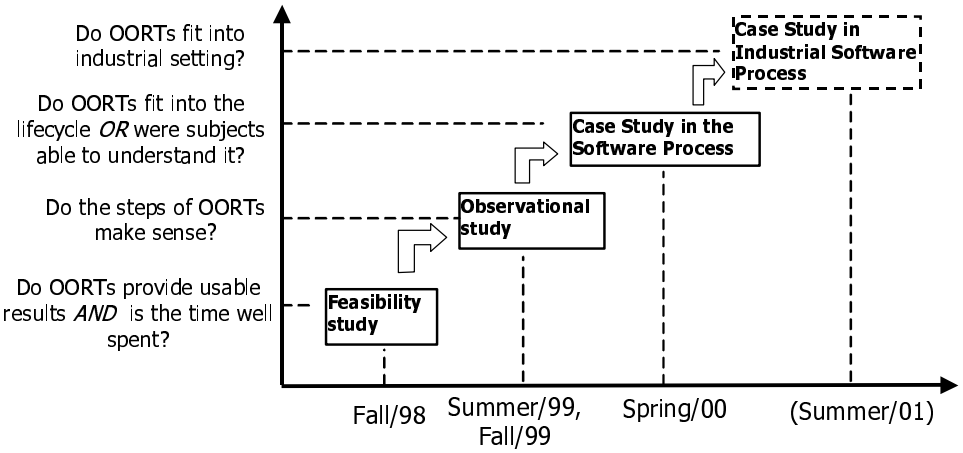

Figure 2: Process evolution of OORTs

a final list that reflected the group consensus.

Data Collection: Questionnaires and interviews were used to collect qualitative data. We also analyzed the artifacts produced during the inspection as a control on the data quality and process conformance. Using both questionnaires and interviews allowed us to collect qualitative data at different times, under different conditions; evaluating the consistency of answers provided a first level of a check on data quality. The qualitative data included:

- Opinion of effectiveness of technique (measured by the percentage of the defects in the document subjects thought they had found)
- Subjective usefulness of different horizontal and vertical reading techniques (open-ended question)
- How closely subjects followed the technique (collected multiple ways for consistency checking)
- Practicality of the techniques (open-ended question)

The questionnaires were also used to capture limited quantitative data, namely the time required for individual review. Analysis of the subjects' defect lists yielded quantitative data concerning the number and type of defects detected by the techniques. (Because this was mainly a feasibility study, we made the assumption in our counting of defects that all defects reported were real problems with the document, rather than spending the time to determine if each reported defect was a real problem with the artifacts. This is a trade-off that can be made during a feasibility study where the main goal is to see if the new process can be used.)

Results and Lessons Learned: The quantitative data from this experiment allowed us to answer the question posed in Section 3.1:

- Using the techniques did allow teams to detect defects (11 were reported, on average).
- Vertical techniques tended to find more defects of omitted and incorrect functionality; Horizontal techniques tended to find more defects of ambiguities and inconsistencies between design documents, lending credence to the idea that the distinction between horizontal and vertical techniques is real and useful.

Thus, the data supported the conclusion that the techniques were feasible: they could be used to detect defects, and moreover could be used to target particular types of defects.

At the same time, the qualitative data showed that in general subjects agreed that the techniques were helpful, and allowed us to answer the question posed in Section 3.2. The data indicated that while the techniques were worth the time that the subjects spent on them, they were not as well specified as they could be. We were able to learn three global lessons on how to improve the techniques:

- OO reading techniques should concentrate on semantic, not syntactic, issues.
- Reading techniques need to include not only instructions for the reader, but some motivation as to why those instructions are necessary.
- The level of granularity of the instructions needs to be precisely described.[2]

These results led us to produce a second version of the techniques that incorporated several global changes, such as a greater focus on semantic checking, more explanation of the goals of the process steps, and a new terminology to help discuss system functionality in more detail.

4.2 Do the steps of OORTs make sense?

Because the previous study had shown us that the techniques were feasible and that the time that it took to use them was well-spent, next we evaluated the steps at a more detailed level. To answer the question from Section 3.3 about whether the steps made sense, we addressed the results from the previous section. This was done using an observational study [15] during the Fall 1999 semester. We also wanted some indication about the problem domains, and the background of inspectors, for which the techniques could be most useful.

Because the observational study was a new approach for us, we first performed a pilot study to debug the observational approach and get it to work in our setting. Only after the pilot study did we perform a full-scale observational study, reported below. The observational study was necessary to understand what improvements might be necessary at the level of individual steps, for example, whether subjects had problems while applying the technique (and how these problems may be corrected), whether each step of the technique contributes to the overall goal, and whether the steps of the technique should be reordered to better correspond to subjects' own working styles.

Subjects: The 14 subjects were members of a graduate-level Software Engineering class. Many were returning professionals; 86% had previous industry experience with OO design.

Materials: The materials consisted of a new version of the OO reading techniques. They were applied to two designs: one for an unfamiliar financial domain (7 classes in the high level design, 4 interaction diagrams and 3 state diagrams) and one for a more familiar parking garage control system (6 classes in the high level design, 5 interaction diagrams and 2 state diagrams).

Procedure: A quasi-experimental, factorial design was used in which half of the class reviewed the design from the unfamiliar domain, and the other half the design from the familiar domain. In each of these groups, roughly half the teams had previously inspected the requirements document for the same system. In this scheme, we could look for any differences in performance due to the reviewers' familiarity with the system requirements or with the problem domain.

Each subject was paired with another student who was trained as an observer. Observers defined their own questions for eliciting

---

[2] For instance, discussing functionality is a difficult but necessary part of the reading techniques. The difficulty comes from the many different levels of granularity at which system behavior can be described, and just assuming that subjects will intuitively grasp the correct level of granularity is naïve and causes frustration for the reviewer.

observational and inquisitive data. After the execution of the techniques, each team wrote an evaluation report discussing their experience and the results of the observation.

Data Collection: We analyzed the artifacts produced by the subjects to collect some quantitative data: the time to execute the techniques and the number and type of defects detected. However, observational techniques were the most important method used in this study. A rich array of qualitative data was collected through their use. The teams produced an evaluation report, which included both a summary of the notes taken during observation as well as retrospective data. The metrics collected from the observations included:

- Users' opinion of effectiveness of technique
- Problems encountered with specific steps of procedure
- How closely the techniques were followed

The retrospective data (collected via open-ended questions) provided the following information:

- Usefulness of horizontal and vertical techniques
- Practicality of the techniques
- The problems found using the techniques

In addition to the data on the individual steps in the process that was provided by the metrics, the retrospective data gave insight into global issues. Also, some of the metrics collected here were the same as in the previous feasibility study, allowing a comparison of results across the two versions of the techniques.

Results/Lessons Learned: The qualitative data gathered during observation provided us with some potential ways of improving the techniques:

- Order of dealing with information (process steps) must match the subjects' own way of thinking about the problem.
- Amount and type of training needed to be modified.
- Differences in design approaches could affect design inspection.

The quantitative data from this experiment allowed us to:

- Verify the difference between types of defects found by horizontal and vertical techniques.
- Show that having domain expertise did not help in the design inspection.
- Show that having been a participant in a requirements inspection for a system did not improve performance in the design inspection.

These results led us to produce a third version of the techniques, using the data from the observations about the way that readers applied the techniques. This version of the techniques also focused more on the semantics behind the design models and less on the syntax. Additional improvements were made regarding training and data collection forms. The details of the process evolution up to this point (along with the third version of the techniques) are presented in [21]. These techniques can be compared with the previous ones presented in [19] to observe the evolution based on these study results.

4.3 Do OORTs fit into a development lifecycle?

Now that we had improved the individual steps in the process, the next step in the methodology was to perform a case study by using the new process inside of a real lifecycle process. This study was done during Spring 2000 semester.

Subjects: The subjects came from a senior level undergraduate software-engineering course. Of the 42 students in the class, 14% had some previous experience with OO design in industry.

Material: The materials under study during this experiment consisted of an evolved version of the techniques based on the results from the previous study. They were applied in the evolution of the familiar domain system used in the previous study.

Procedure: The subjects used a waterfall development process, to create an enhanced version of an existing system.

Once the initial design had been created, all teams used the horizontal reading to inspect their own designs. After correcting any defects, each team then performed the vertical reading techniques on a design for another team.

In the overall scope of the software development process there was no control group. This lack of a control group occurred for two reasons: first, the design inspection was one small part of a larger experiment, and the overall experimental design did not allow for a control group. Secondly, it was in a classroom environment.

Data Collection: Questionnaires and analysis of created artifacts were used to evaluate the effectiveness of the techniques in the development process. The questionnaires were used throughout the development cycle to collect both qualitative and quantitative data. The quantitative data collected include both background information and the amount of time taken to use the techniques, used to evaluate the use of the techniques in the lifecycle process. The qualitative data collected by the questionnaires concerned:

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

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

Results/Lessons Learned: The qualitative data from this, our first case study, provided us with some lessons about the techniques and how they fit with other development processes. First, we found that subjects were able to apply the techniques inside of a lifecycle and in combination with other processes (specification, design, implementation, and testing). The techniques were useful for inspections, as they resulted in defects that were corrected to improve system quality; but, in the context of system development they also turned out to have another use: The vertical techniques helped students to gain a better understanding of the system functionality and how it should be represented in the design. Also, the techniques were feasible to

use during development, as the effort required was not prohibitive compared to other system tasks; the design inspections required on average 20 hours per team, or 24% of the overall effort spent on design. Outside of the training in the techniques, the subjects required no special knowledge that was not previously gained during the development of the system.

4.4 Do OORTs fit into an industrial setting?

At this point in time, having run a case study in a classroom environment, our next step is to run an industrial case study to make sure these ideas can be tailored and transferred to an industrial environment. For this study we are currently seeking an industrial partner. The series of studies run to date has provided a body of evidence that, first, yielded a proof-of-concept of the usefulness of the process and second, identified a set of issues that we know will be important for tailoring this process for effective industrial use. For example, we have already observed the effects of subject training and of previous subject experience (with inspections in general, with the problem domain, and with OO concepts) and how they can influence the effectiveness of the process.

Every pilot study of a new process in an industrial environment needs a back-out plan, i.e. a way of responding with minimal disruption to the success of the project if the process turns out to be ineffective in the environment. However, the studies run to date have formed a responsible approach for developing confidence in the process under study and for demonstrating that we can tailor it to the industrial environment.

5 CONCLUSIONS

In this paper, we have outlined an approach for evolving processes, from the early concept phase to the tailoring and use of the process on an industrial project. We have illustrated how a body of evidence concerning process effectiveness can be built up, using different types of studies to address different questions of interest about a process. We believe that such an approach is helpful for a responsible interaction with industry.

For researchers, we have provided a methodology for planning such an iterative approach to evolve and study processes. We have given some indication, based on our own experience, of heuristics for deciding what type of study and what type of data collection is best suited for a given stage of process evolution.

For practitioners, we have discussed a process for addressing an important development task, OO design inspection. We have illustrated the series of studies used to build up confidence in the process and understand the relevant variables, so that readers can understand and judge for themselves the evidence regarding the process' effectiveness. We have argued that the process can continue to be adapted, adopted, and studied in an industrial environment without undue risks to the project.

We invite interested readers to our web page, www.cs.umd.edu/projects/SoftEng/ESEG/manual/OORTs/, which contains pointers to information on the studies referenced in this paper, a particular OORT at various stages of evolution, and further references.

ACKNOWLEDGEMENTS

This work was partially supported by UMIACS and by NSF grant CCR9706151. We wish to thank Prof. Victor Basili for his support and our subjects for their hard work. Dr. Travassos also recognizes the partial support from CAPES-Brasil.

REFERENCES

[1] Basili, V. R. and Caldiera, G. "Improve Software Quality by Reusing Knowledge and Experience," Sloan Management Review 37, 1 (Fall 1995), 55-64.

[2] Basili, V.R., Green, S., Laitenberger, O., Shull, F., Sorumgaard, S.L., and Zelkowitz, M.V. "The Empirical Investigation of Perspective-based Reading." *Empirical Software Engineering, An International Journal*, Volume 1, Number 2, pp. 133-164, Kluwer Academic Publishers, October 1996.

[3] Basili, V.R. "Evolving and Packaging Reading Technologies." Special Issue, *The Journal of Systems and Software*, Volume 38, Number 1, pp.3-12, July 1997.

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

[5] Boehm, B. and Basili, V. "Software Defect Reduction Top 10 List", IEEE Computer, 34, 1, (January 2001) 135-137. www.cebase.org

[6] Campbell, D.; and Stanley, J. *Experimental and Quasi-Experimental Designs for Research.* Houghton Mifflin Company, Boston 1963.

[7] "Concluding Panel: Metrics Faceoff – What Industry Needs from Researches: What Researchers need from Industry." Software Metrics Symposium, Bethesda, MD, Nov 1998.

[8] Höst, M.; Regnell, B. and Wohlin, C. Using Students as Subjects: A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. In Empirical Software Engineering – An International Journal. Vol. 5, No. 3, Nov 2000

[9] Kitchenham, B.; Pickard, L.; and Pfleeger, S.L. Case Studies for Method and Tool Evaluation. *IEEE Software* 12, 4 (July 1995), 52-62.

[10] Pfleeger, S.L. *Software Engineering: Theory and Practice*. Prentice-Hall, 1998.

[11] Porter, A.; Votta Jr., L.; and Basili, V. R. Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment. *IEEE TSE* 21, 6 (June 1995), 563-575.

[12] Pressman, R. *Software Engineering: A Practitioner's Approach*, 4th ed., McGraw-Hill, 1997.

[13] Rombach, D. "Fraunhofer: The German Model for Applied Research and Technology Transfer." In *Proc. ICSE'00* (Limerick, Ireland, Apr. 2000), 531-7.

[14] Shull, F. *Developing Techniques for Using Software Documents: A Series of Empirical Studies.* PhD Thesis, Computer Science Dept., University of Maryland. 1998.

[15] Shull, F.; Travassos, G.; Carver, J.; and Basili, V. R. *Evolving a Set of Techniques for OO Inspections.* University of Maryland Technical Report CS-TR-4070. October 1999.

[16] Shull, F.; Rus, I.; and Basili, V.R. How Perspective-Based Reading Can Improve Requirements Inspections. *IEEE Computer* 33, 7 (July 2000), 73-79.

[17] Singer, J.; and Lethbridge, T. Methods for Studying Maintenance Activities. In *Proc. of the Workshop for Empirical Studies of Software Maintenance* (Monterey CA, Nov. 1996), 105-110.

[18] "Software Past, Present, and Future: Views from Government, Industry, and Academia." Panel Discussion, NASA Software Engineering Workshop, Greenbelt, MD, 1999.

[19] Travassos, G.; Shull, F.; Fredericks, M.; and Basili, V. R. Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality. In *Proc. OOPSLA'99* (Denver CO, Nov. 1999), ACM Press, 47-56.

[20] Travassos, G.; Shull, F.; Carver, J.; and Basili, V. R. Reading Techniques for OO Design Inspections. In *Proc. of the 24th Annual Software Engineering Workshop* (Greenbelt MD, Dec. 1999), NASA Goddard Space Flight Center (SEL-99-002).

[21] Travassos, G.H., Shull, F. and Carver, J. "A Family of Reading Techniques for OO Design Inspections." In Proceedings of the WQS'2000 – Software Quality Workshop, (October 2000), Brazilian Symposium on Software Engineering, p.225-237.

[22] Werner, C.M.L., Travassos, G.H., Rocha, A.R.C. "An OO Software Engineering Training Experience within a Collaboration Project between Academia and Industry." In *Proceedings of TOOLS27 – Technology of Object Oriented Languages and Systems*, IEEE Computer Society, pp.290-294, Beijing, China, 1998.

[23] Zelkowitz M. V. and D. Wallace, Experimental models for validating computer technology, *IEEE Computer* 31, 5 (May, 1998) 23-31

[24] Zhang, Z.; Basili, V. R.; and Shneiderman, B. An Empirical Study of Perspective-Based Usability Inspection. In *Proc. of Human Factors and Ergonomics Society Annual Meeting* (Chicago IL, Oct. 1998).