Understanding the Impact of Assumptions on Experimental Validity

Jeff Carver[1] John VanVoorhis[1] Victor Basili[1, 2]

[1]*Department of Computer Science*
University of Maryland
{carver, jvv, basili}@cs.umd.edu

[2]*Fraunhofer Center for Experimental*
Software Engineering
Maryland

Abstract

Empirical studies are used frequently in software engineering as a method for studying and understanding software engineering techniques and methods. When conducting studies, researchers make assumptions about three objects, people, processes and products. Researchers usually focus their study on only one of those objects. But, regardless of which type of object is chosen as the focus of the study, researchers make assumptions about all three objects. The impact of those assumptions on experimental validity varies depending on the focus of the study. In this paper, we discuss the various types of assumptions that researchers make. We relate those assumptions back to some concepts from social science research. We then use the results of a people-focused study to illustrate the impact of the assumptions on the results of that study.

1. Introduction

Empirical studies are useful for understanding software engineering techniques, but to draw accurate conclusions from a study, researchers must account for factors that could cause those results to be misinterpreted. Competing explanations for the results must be eliminated, where possible, so that researchers can have greater confidence in their conclusions. Threats to validity are issues that can call into doubt the results of a study or the conclusions drawn from those results. By understanding and properly addressing threats to validity, a researcher can make the results of a study more widely useful and applicable.

When accounting for threats to validity in their study design, researchers must make assumptions about their environment. If any of these assumptions prove to be false, then it is more difficult to properly interpret the results of the experiment. In this paper, we identify a set of common assumptions and then investigate their impact on the validity of a study.

Software Engineering Experimentation

Experimentation has been found to be an effective research tool in software engineering. A growing number of researchers are including experimental results in their research work [19]. Experimentation allows researchers to test hypotheses they have formulated about a phenomenon. The results of empirical studies provide researchers with hard evidence to back up their conclusions.

While there are many benefits to experimentation, there are also some drawbacks, especially when human subjects are involved. The planning of software engineering experiments requires a large amount of effort from the researcher. Furthermore, software developers' time is expensive, so the cost of running a software engineering experiment is high. Statistically significant results are difficult to obtain from small sample sizes. Another difficulty is the individual variations in human subjects. In any one study, the variation among the subjects can outweigh the influence of the real variable of interest [11, 12, 14].

Yet, to advance the state of knowledge in the field, it is necessary to conduct and replicate experiments. As Basili, et al. have pointed out, the hallmark of good experimentation is the accumulation of data and insights over time [3]. This accumulation of knowledge and insight also allows researchers to investigate complex questions. As more data about a software engineering process is collected, researchers can identify and study other variables that may affect its use [6]. The level of analysis described in this paper is only possible after conducting and replicating multiple experiments on a software engineering process.

Threats to Validity

Software engineering researchers must identify and address various issues that may call into question the validity of their results. Threats to *internal validity* are those issues that cause the conclusions drawn from the data to be questioned. Conversely, threats to *external*

validity are issues that call into question the applicability of the conclusions to other environments [4]. Judd, et al, identify *construct validity* as the extent to which the concrete measures in the study successfully duplicate the theoretical constructs in the hypotheses [9]. Cook and Campbell pose another type of validity to consider: *conclusion validity* [8]. Basili, et al, discuss these threats to validity and point out some difficulties that arise when trying to obtain all types of validity in the same study [3].

Lying underneath these types of validity are some basic, often unstated, assumptions about the people, processes, and products in the study. The impact of these assumptions will vary depending on the focus of the study. During the study design process, researchers must balance the various threats to validity and the assumptions to create the best design possible.

2. Common Assumptions

Software engineering experiments are similar both to traditional computer science experiments and to experiments in the social sciences (e.g. psychology). In traditional computer science experiments, researchers tend to study the effects of some *process* (e.g. an algorithm) on some *product* (e.g. a set of data). In the social sciences, experiments tend to study a *person* using a *process* to accomplish some task. Therefore, traditional computer science experiments and social science experiments typically have to account for only one class of object in addition to the process being studied.

In contrast, software engineering experiments often study a *person* using a *process* on a *product* so they have to account for all three objects. In order to design valid experiments, software engineering researchers should be informed by both physical science and social science experimentation methods.

When all three objects (people, products and processes) are involved in a study, researchers will make a series of assumptions about each one. These assumptions will vary with the focus of the study, but many remain constant across all types of studies.

People

Software engineering experiments often focus on human subjects [1,18]. Mental states, attitudes, skills, and individual knowledge are not directly observable, so measuring these characteristics or even knowing what should be measured can be a difficult task. Several psychology and sociology journals, such as the *Journal of Applied Measurement*, *Rasch Measurement Transactions*, *Psychometrika*, the *Journal of Outcome Measurement*, and the *Journal of Educational Measurement*, are devoted to these types of measurement concerns.

To study the differences among groups of people, researchers must characterize those subjects and form homogeneous groups. Often, subject characterization is done using questionnaires. These data are self-reported, so their reliability and validity may be suspect.

Two major assumptions made about people are:

1) The measures used to characterize the people are valid.
2) The characterization data (questionnaires) provided by the subjects is reliable and valid.

Processes

When a new process or technique is being studied, normally a training session is conducted to give all the subjects the same level of exposure to the process. This training is assumed to provide the subjects knowledge and skill. *Knowledge* of a technique means that the subjects understand the goals and operations of a technique and can answer questions about it. *Skill* means the subjects are able to effectively use a technique with some level of expertise. When conclusions are drawn about the use of a technique, researchers assume that the subjects have followed the technique.

Two major assumptions that researchers make in their studies about processes are:

1) Training given to the subjects is adequate.
2) Subjects follow the process they are given.

Products

Many real-world software artifacts are too large to be used in a study, so researchers must either scale them down or create "toy" problems. Researchers have to face the question of whether the conclusions drawn based on one set of artifacts can be generalized to other artifacts. For a given artifact the application domain, scope of the information, and seeding of the feature being measured, such as defects must be considered.

Three major assumptions that researchers make about the products are:

1) Format of the artifact is appropriate.
2) Scope of the problem is sufficiently complex.
3) Feature being measured in the product was accurately represented.

Any one of these three objects – people, processes, or products – can be the focus of a software engineering study. The impact of the assumptions about each object varies based on which object is the focus of study. However, regardless of which object is chosen, the various threats to validity must be balanced.

The results of a study with *external validity* hold for other populations, e.g. industrial software engineers. A study having *internal validity* means that the observed changes in performance, e.g. effectiveness at some task, can accurately be attributed to the treatments in the study.

A study having *construct validity* means that the observed effect is the result of a theorized cause, e.g. the operationalization of a theorized process has a positive effect on the outcome.

3. Study Foci

A **process-focus study** primarily addresses questions about a software process. For example: "Is Technique A more effective than Technique B for accomplishing a given task?" A **people-focused study** primarily addresses questions about people involved in software development. For example: "What effect does some characteristic of a subject have on his or her effectiveness using a technique?" Finally, a **product-focused study** primarily addresses questions about a software product. For example: "For a given product, does Format A or Format B allow the subject to accomplish a given task more effectively?"

Many of our past studies have been process-focused studies. In a process-focused study, the people and product assumptions generally affect the external validity of a study. For example, improperly characterizing a subject will typically not affect the conclusion about the relative effectiveness of two techniques. Conversely, a suspect subject characterization may affect the external validity of the study, by making the results of the study not generalizable. Likewise for the product assumptions, an unrealistic artifact will likely not affect the conclusions drawn about the process, but it could affect the applicability of the results to industrial artifacts.

In a product-focused study, e.g. a study of a requirements document, the product assumptions affect the internal validity of the study because conclusions drawn about the artifacts depend on those assumptions. These product assumptions also affect the construct validity because the artifact should challenge specific skills or abilities of the subject in ways that accord with the given theory of the problem domain. The people assumptions and process assumptions will tend to impact the external validity. The people assumptions impact the generalizability of the results to other groups of people. The process assumptions also affect the applicability of the results to other processes.

In this paper, we address the impact of process, product, and people assumptions on a people-focused study; thus, the domain of social science becomes relevant, and insights from social science researchers helpful. In psychology and sociology, a *construct* is an object of study that is not directly observable or measurable. The object of study is a theoretical concept that is not observable itself. In the social sciences, researchers focus on understanding the relationship between a mental construct and a concrete representation of that construct, such as a document or process.

Figure 1 describes the relationships between theoretical constructs and physical representations, or operationalizations, of those constructs. Theories represent the predicted relationship between a cause and effect. In an experiment, a theoretical cause construct is operationalized into some treatment (program) and then applied by the subjects. The theoretical effect constructs are then operationalized into the metrics (observations) that are collected in the experiment itself.

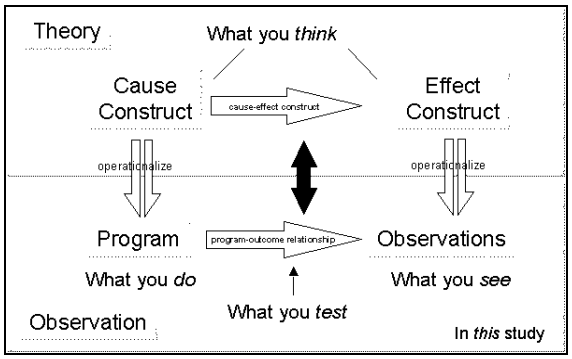

Figure 1 –Theoretical Constructs Used in Experimentation [17]

We can use Figure 1 to indicate the source of different threats to experimental validity. The arrows represent inferences (assumptions) made by the researcher that may or may not be valid in a given study. Arrows from *theory* to *observation* (experiment) represent the source of potential construct validity problems. That is, were the *program* (experimental treatment) and the *observations* (experimental metrics) an accurate operationalization of the theoretical *cause* and *effect constructs*? Likewise, the arrow from *program* to *observation* represents the source of potential internal validity problems. That is, were the *observations* really produced by the *program* (experimental treatment)?

The assumptions in Section 2 arise from the potential mismatches between theoretical concepts and concrete representations. The assumption that experience measures are valid (*people*) is really an assumption that the theoretical construct, experience, was accurately operationalized into a concrete set of experience metrics. For example, is amount of experience (number of projects) an accurate representation of expertise (task proficiency)? The assumption that the training was adequate (*process*) is really an assumption that the training lecture was an accurate representation of the base knowledge needed to effectively use the process. For example, did the training lecture focus spend the appropriate amount of time on each topic? The assumption that the artifact format was appropriate (*product)* is really an assumption that the concrete representation of the author's mental model of the domain (the format of the artifact) allowed the reader of the

artifact to process information more easily than another representation would. For example, does breaking up user-level functions into smaller units help in understanding the artifact?

For each of the three objects, an important consideration is balancing the various threats to validity. Increasing the internal validity often reduces the external validity. Controlling internal validity too much, at the expense of external validity, can affect the construct validity of the experiment. For example, if a researcher constrains the domain of the artifacts too much, the subjects' prior knowledge of the world may conflict with the arbitrary bounds and constraints of domain of the artifact. In this situation, construct validity is challenged because the mental models formed by the subjects may be different from those assumed by the researcher.

Therefore, there is a tradeoff between internal validity, external validity, and construct validity. A researcher must typically choose to emphasize one of the three types, but has to also consider the other two types during the planning of a study. As the study focus changes, and with it the impact of the assumptions, these tradeoffs must be reassessed to optimize overall validity.

Section 4 describes a people-focused study. In our previous studies, there was some variation in the individual performance of the subjects, so by running a people-focused study we hoped to better understand that variation. We did not fully understand the impacts of changing the study focus from process to people until the conclusion of the study. Section 5 describes the impact of the assumptions on the validity of the study.

4. The Study

The main goal of this study was to understand the effects of reviewers' experience on process use. In changing from a process-focused study to a people-focused study we faced new questions about the impact of the assumptions. This section provides an overview of the study necessary for the discussion of assumptions in Section 5. We provide a complete discussion including all of the results elsewhere [7].

4.1 Background

The selection of inspectors based on individual characteristics, such as software development experience and domain experience, has an impact on the defects found in an inspection. It has been suggested that a detailed requirements inspection technique might neutralize the defect detection benefit of development experience [5]. Jose Maldonado hypothesized, based on a study conducted in Brazil, that if a technique is too detailed, and an experienced developer follows the technique, he or she may find fewer defects than with a less detailed technique that allowed him or her to use their expertise [J. Maldonado, personal communication, July 2002].

To address these issues, this study was run to understand the interaction between an inspector's software development experience and the amount of detail in their inspection technique. The overall hypothesis of the study was:

> *To be effective, the level of detail in a technique must be tailored based on the inspector's experience. More experienced inspectors need less detail while less experienced inspectors need more detail.*

To evaluate this hypothesis, we had to accurately measure the **people** (subjects' experience), the **process** (the quality of their training in the technique), and the **product** (the appropriateness of the inspected artifact, its defects, and their distribution).

4.2 Study Design

4.2.1 Subjects. The subjects of the study were 22 students in a graduate level, software engineering class at the University of Maryland.

4.2.2 Materials. The subjects inspected a requirements document for a Parking Garage Control System (PGCS) for managing a parking garage. The system was responsible for keeping track of monthly and daily (pay for each use) parking tickets and only allowing cars to enter the garage if there was space available. The PGCS requirements document had 17 pages, which included 21 functional and 9 non-functional requirements. There were thirty-two defects, some were seeded by the experimenters and some occurred naturally.

Each subject used one of two versions of Perspective Based Reading (PBR), a requirements inspection technique. In PBR, each inspector assumes the perspective of one of the requirements stakeholders, e.g. end user, designer, or tester. The inspector then creates a model of the requirements, which is an abstraction that is relevant to the stakeholder represented by that perspective, e.g. a tester would create a set of test cases. The inspector also is given a series of questions to answer to help uncover defects [2].

The first version of PBR was a **high-level version** that provided the inspectors with general guidelines about creating the abstraction model. It did not instruct the inspector on *how* to construct the model but relied on the inspector's software development expertise to create the model. The second version was a **step-by-step version** that provided detailed instructions for creating the abstraction model. It relied less on the inspector's software development expertise.

4.2.3 Procedure. The subjects were initially given a questionnaire to gather their background and experience in different software development tasks relevant to the PBR perspectives and in different application domains. Because there were not enough subjects to use more than one perspective, the results of the questionnaire allowed us to choose the *tester* perspective as the one with the best ratio of experienced to inexperienced subjects.

We developed a criterion, *a priori,* for splitting the subjects into two experience groups. We chose the criterion that high experience subjects (Group H) would be those with experience testing on at least one industrial project and low experienced subjects (Group L) would be those with industrial testing experience on either 0 or 1 project. The responses to the background questionnaire placed approximately 1/3 of the subjects were in Group H and the other 2/3 in Group L.

Table 1 – Experimental Groups

Group	**H**	**L1**	**L2**
Experience	High	Low	Low
PBR Technique	High-level Version	Step-by-Step Version	High-level Version
Abstraction Model	Not Provided	Provided	Not Provided
Number Subjects	6	10	6

Table 1 shows how the techniques were assigned to the inspectors. The high-experience subjects (Group H) were all given the high-level version of PBR, because they were able to rely on their expertise to create the test cases. There were twice as many low-experienced subjects (Group L), so they were divided into two groups (Groups L1 and L2). Subjects in Group L1 received the step-by-step version of PBR, because they were not able to rely on their expertise to create the test cases. Subjects in Group L2 received the high-level version of PBR so they could act as a control group.

The training covered two class periods. On day one, the subjects received an hour of training on inspections, defects, and reading techniques in general. On day two, all of the subjects received 30 minutes of training on the goals and theories of PBR. Finally, the subjects received another 20-30 minutes of training on their assigned version of PBR.

4.2.4 Data Collection. While performing the inspection, the subjects recorded defects on a standard defect report form. At the completion of the inspection, the subjects were given two questionnaires where they could discuss their experiences and problems using PBR and anything they had learned by conducting the inspection.

4.3 Results

The hypothesis posed in Section 4.1 was decomposed into three testable hypotheses.

H1 The subjects from Group H would be more effective and efficient than the subjects from group L2 (the control group).

This hypothesis was testing whether the expertise of the subjects in Group H would result in improved performance over the subjects in Group L2.

H2 Subjects from Group L1 would be more effective and efficient than subjects from Group L2.

This hypothesis was testing whether, for low experienced subjects, the increased detail in the technique used by the subjects in Group L1 would improve their performance over the subjects in Group L2.

H3 Subjects from Group H would be more effective and efficient that high experience subjects from a previous study who used the step-by-step version of PBR.

This hypothesis was testing whether, for high experienced subjects, less detail in the technique used by the subjects in Group H would results in improved performance over the subjects from a previous study.

Table 2 – Raw Data

Group	**H**	**L1**	**L2**
Defect Rate	21.6%	20.6%	26.5%
Efficiency (defects/hour)	4.9	2.6	4.2

Table 2 shows that the data do not support H1 and H2. The subjects from group L2 were more effective than both the subjects from H and the subjects from L1. The subjects from H were more efficient than the subjects from L2, but the difference was not significant.

To evaluate H3, we used the data from a previous study in which high experienced subjects inspected the PGCS requirements document using a more complex step-by-step version PBR than was used in this study. In that study, the average defect rate was 26.7% with an efficiency of 1.9 defects/hour. In terms of effectiveness, the data does not support H3. Conversely, the subjects from group H were significantly more efficient than the historical subjects (4.9 defects/hour).

Based on our previous experience we expected the data to support, rather than contradict our hypotheses. To understand why our expectations were not met we performed a root cause analysis. We chose to focus our analysis on the assumptions because our expectations were based on the veracity of those assumptions.

5. Discussion of Assumptions

Many of the human experience characteristics were not directly measurable, so it was necessary to measure them indirectly. This indirect measurement caused several assumptions to depend on the reliability and validity of the measures. *Reliability* is the consistency or repeatability of a measurement (i.e. the likelihood that the same results will be obtained if the measure is repeated). *Validity* is the accuracy of a measurement (i.e. the proximity to the true value) [17].

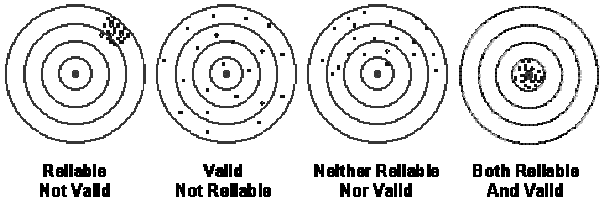

Figure 2 – Measure Validity and Reliability [17]

Figure 2 illustrates the difference between reliability and validity. The center of the bull's-eye is the true value the measurement tries to obtain. The dots represent actual measurements. A cluster of measurements indicates a reliable measure, because successive measurements obtained similar values. A set of measurements with an average that is close to the true value indicates a valid measure that is likely to provide an accurate value [17].

Defect detection rate is likely a valid measure of defect detection ability, but it is not very reliable. It can fluctuate based on several factors other than a subject's ability. Alternatively, a self-reported number of years of experience in testing is likely a reliable measure – subjects should give the same answer if asked the question again. Yet, it is not clear that it is a valid measure of true testing skill. The *quantity* of experience (number of years) may not necessarily translate into the *quality* of experience (expertise).

As the focus of our experiments shifted from processes (e.g. [2, 5, 16]) to people, the assumptions discussed in Section 2 begun to impact different types of threats to validity. In this section, we will discuss the assumptions made about people, products and processes and their impact on the validity of the study.

5.1 People

There were two assumptions made about people. These assumptions arise frequently when dealing with human subjects.

People_1) Our experience measure and grouping criterion were valid.

People_2) The amount of experience reported by the subjects was reliable and valid.

5.1.1 Assumption People_1. *People_1* impacts the construct validity of the study. Our theoretical concept stated that experience as a tester would improve the effectiveness of an inspector. That concept of experience was not directly observable, so it was operationalized it into observable metrics. The operationalization had two factors: location (industry vs. school) and amount (number of projects). Specifically, we assumed that someone with experience in industry was more experienced than someone with experience in the classroom and that a greater number of projects indicated more experience than a smaller number of projects.

The study results showed little difference in the effectiveness of subjects with low testing experience and the subjects with high testing experience. In fact, the subjects from Group L2 were the most effective of all the subjects. This result cast doubt on the veracity of our assumption about experience.

The qualitative data from the study also indicated that our measure of testing experience and our criterion for grouping did not seem to correspond to our theoretical concept of experience:

1) Of the six subjects in Group H, two stated that the training needed more examples of creating test cases. Four stated that there were not enough details in the technique.
2) All of the high experience subjects reported that they needed additional background, which was not included in the lecture, in order to understand and use the technique.

These two results would not be expected from experienced testers so we concluded that the subjects who we classified as having high experience might not have had high experience. Possible reasons for the misclassification of subjects are the metric "number of projects" was not a valid operationalization of expertise, our criterion for grouping subjects was not a valid break point along the theoretical continuum of experience, the type of testing experience possessed by the subjects did not match up with that needed for the exercise, or we simply did not capture the right kind of experience.

We believe that experience is important, but it is not always clear how to operationalize experience into a metric. For example, which experiences are truly important? Is number of projects really the correct measure? We are interested in a subject's ability to effectively perform a particular task, not necessarily how long they have been doing that task. Thus, one subject might become proficient in a day, while another subject might not become proficient even after years of experience. In this study, the amount of experience was not a good proxy for proficiency.

5.1.2 Assumption People_2. The potential for lack of accuracy in self-reported data arises from two sources. First, when asked to rate themselves, people often overestimate their experience. In a recent study, the correlation between self-reported ability and tested ability was less then 20% [13]. Second, some questions can be interpreted in multiple ways. Thus, two subjects with the same actual experience level might rate themselves differently.

The subjects were asked to report their experience in various aspects of software development at the classroom and industrial level. The definition of "experience" was ambiguous and open to interpretation. For example, does testing experience on one project in industry mean that the subject was the main tester for that project or that the subject performed some minor testing activities on one project? These two interpretations are very different.

Application domain knowledge was also obtained through self-reported data. On a post-experiment questionnaire, the subjects were asked to give examples of places where the information in the requirements was new or different from what they expected. Two items seemed to make a difference on defect detection effectiveness, the inspector's mental model of a parking garage, and the model presented in the requirements. The average defect rate for the six subjects who stated that their mental model of a parking garage was in conflict with the model in the requirements document was 18.8%; while the average defect detection rate of the seven subjects who said the requirements document helped them clarify their mental model was 25.5%. This difference was not statistically significant, but suggests a potential influencing factor for defect detection rate.

When the inspected artifact operationalizes the domain information in different way than the inspector's theoretical model, there seems to be difficulty during the analysis. When the subject's theoretical model of the domain is clarified by the operationalization, then the inspector seems to do well. This result may reflect either better synthesis of the new information by the subjects due to innate ability, or an inherent problem with new or different domains that inhibits detection ability.

5.1.3 Tools to address assumptions. To reduce the uncertainty in these assumptions we recommend using more objective measures of subject experience. One potential method is using a pre-test and/or post-test. Instead of using self-reported experience, a pretest or posttest could allow subjects to be grouped based on a more objective evaluation of their experience. Furthermore, this objective test could also help account for the secondary variables. Another recommendation for addressing this assumption is better validation of the background survey [10]. By accurately characterizing the subject population, researchers can determine the likelihood that confounding factors are present.

Overall, the people assumptions impacted the internal and construct validity of our people-focused study. False assumptions about subject experience have forced us to question the accuracy of our study results. We need to develop better methods for collecting the true experience of subjects.

5.2 Processes

There were two assumptions made about the process:

Process_1) The training was adequate for the subjects to become competent in using PBR.

Process_2) The subjects followed the PBR process.

5.2.1 Assumptions. These assumptions impact two threats to validity, internal and construct. The threat to internal validity is present based on the assumption that the training was effective. One technique might be ten times more effective than another, but if the training is ineffective, and the technique is not used properly, that difference may not be seen. The threat to construct validity of the process is present because a training lecture can provide knowledge of the technique, but it is more difficult to guarantee that it provides skill development in the technique. The training is the operationalization of the technique as a theoretical construct. Furthermore, it is also difficult to ensure that subjects who were adequately trained actually follow the process.

Overall, the subjects indicated that the training was adequate, with only one subject from each of the three treatment groups indicating that the training was insufficient. However, the subjects did indicate some desired improvements in the training. Half of the subjects who were trained in the step-by-step version of the technique needed more examples to better understand the details of the technique. One-third of the subjects trained on the high level version of the technique needed more details about creating test cases. Furthermore, many of the subjects said that they needed additional knowledge they did not possess, e.g. about testing in general, about the specific abstraction model used in PBR, and about the application domain. While self-reported experience levels may be unreliable (see Section 5.1.3), self-reported observations are a valid representation of a person's opinion.

The researcher must decide whether the overall goal of the training is to provide the subjects with an understanding of the theory behind the technique and its applicability; or to equip the subjects to effectively use the technique. The two contrasting goals require different training approaches. If the goal is for the subjects to

acquire knowledge about a technique, then classroom lectures including theory and proofs of correctness are more appropriate. If the goal is for the subjects to be able to effectively use the technique, then more laboratory sessions where the subjects can practice the technique and receive expert feedback should be provided.

5.2.2 Tools to address assumptions. Once the goal of the training has been identified, then Proc_1 can be addressed. One method for addressing this type of assumption is use of a pre-test/post-test design. The knowledge and skill of the subjects can be measured prior to training and then again after training to determine what they have learned. Conversely, an evaluation test could be created that would rate the knowledge and skill level of each subject based on an objective measure. Finally, subjects could be kept in the training process until they reach a certain level of knowledge and skill. Whichever method is chosen, the goal is to know, rather than assume, the impact of the training.

5.3.3 Evaluation of tools. In this study, we piloted the use of a more objective measure to look at the effectiveness of our training. We developed two requirements excerpts that could be used as a pretest and a posttest for the subjects to complete before and after the training session. The goal was to measure the improvement of the subjects from the pretest to the posttest.

We attempted to make the pretest and the posttest comparable. We identified three types of requirements, one that had an obvious defect, one that had a subtle defect, and one that had no defect. We placed one requirement of each type in the pretest and in the posttest. Furthermore, we gave the pretest and the posttest to an independent reviewer to help us determine if they were of approximately equal difficulty.

Table 3 – Pretest/Posttest Results

Req. Type	Excerpt 1	Excerpt 2
No Defect	24%	20%
Obvious Defect	30%	90%
Subtle Defect	44%	37%

Despite these efforts, as Table 3 shows, the percentage of subjects that correctly identified each type of requirement for each of the two requirements excerpt were not equal. We underestimated the effort necessary to create a valid pretest and posttest. Our experiences point out the difficulties involved in creating valid pretests and posttests for measuring the effectiveness of the training.

5.3 Products

There were three assumptions made about the products:

Product_1) The format of the artifact was useful.
Product_2) The defects were reasonably independent and a good representation of actual defects.
Product_3) The scope of the problem was realistic.

Prod_1 and Prod_2 are related to construct validity. The format of the artifact is assumed to be a rational operationalization of the author's theoretical model of the domain. If the format of the artifact is too different than what is normally seen in industry, additional validity problems could arise. A defect is a construct of a theoretical belief of an error in a software artifact. The scope of the problem is related to the external validity of the study.

5.3.1 Assumption Product_1. This assumption arises from the fact that the requirements were formatted such that the steps necessary to achieve a user level function were broken up into smaller functional units. Therefore, several numbered requirements must be read together to understand a single function. Furthermore, the artifact did not provide a description of the system structure. The results of the study indicated that those subjects who felt that the format was helpful were less effective in finding defects than those who saw the format as problematic. These results echo the divergence between subjective assessment and objective reality found in the Cleanroom experiments [15].

5.3.2 Assumption Product_2. This assumption arose because the results of this study rely on the percentage of defects found. One method for understanding the defect quality is to obtain feedback from the subjects. At the conclusion of the study, the subjects were provided with the master defect list and asked to indicate whether they agreed or disagreed that each item on the list was a true defect, and to explain why. There were some disagreements as to what constituted a true defect. One of the interesting results was that we also asked the subjects to indicate which of the defects on the master list they believed they reported during their inspection. There were many cases where the subjects indicated they had found a defect that the experimenters had not given them credit for and vice versa. The results of this exercise further indicate the mismatch between the constructs of the experimenters and those of the subjects. Both the parking garage concepts and the requirements defects concepts were different among the subjects.

5.3.3 Assumption Product_3. The third assumption arose because the scope of the "toy" problems that must be used in this type of study is artificially smaller than it would be in the real world. Qualitative data was collected to understand the subjects' opinion of the scope of the inspected artifact. Over half of the subjects found something new or unfamiliar in the document showing that the domain was not as familiar as we had assumed. Like people and their mental models, textual documents also have unobserved attributes. A defect is an unobservable attribute of a software artifact because the reviewer must make an inference using their existing knowledge and the text to find a defect.

The subjects were evenly divided in their opinions whether the scope of the requirements was well defined and whether there was enough information in the document to continue to design and build a system. The defect detection rate for each group reveals a more interesting result. Those subjects who felt the requirements did not adequately describe the scope of the project found fewer defects than those who felt the scope was well defined. This result suggests that subjects who were less certain of system boundaries might have assumed that information left out was outside of the scope and therefore not a defect. More interestingly, those who felt that there was not enough information present to start the design found more defects than did those who felt there was enough information. It is not clear whether the subjects understanding of the problems with the requirements caused this result or if it was caused by some other factor.

5.3.4 Tools to address assumptions. In addition, there are several other avenues to pursue in characterizing and evaluating artifacts. First, any experiment involving an artifact as complex as a requirements document, even for a "toy" problem, must be pilot tested. Second, if multiple artifacts are used in a study, then pilot studies should be run to ensure that the defects seeded in each artifact are comparable. Third, requirements defects are more subjective than code defects, so researchers should remember that a defect is an operationalization of a theoretical concept. Finally, other issues that can affect construct validity include overall document size, defect density, defect realism, and defect type distribution. Thus, a proper operational definition (or a correct defect seeding) is critical for producing valid conclusions about defect detection.

6. Conclusions

Table 4 summarizes the lessons learned about assumptions. These lessons are extracted from the discussion in Section 5. Each lesson is characterized based on whether it applies to People, Products, or Process or to more than one object.

Table 4 – Lessons Learned

Object	Lesson Learned
People	The *amount* of experience is not a good proxy for *expertise*
People	Experience should be objectively measured, rather than subjectively self-reported
Process	Subjects should be objectively evaluated after training to ensure competence
Product	If the measurement objective (such as defect) is not well defined, it can be interpreted differently by experimenters and subjects.
Multiple	A large amount of efforts is required to create valid pretests and posttests

When1 designing software engineering studies, it is critical for researchers to add a step to their process in which they identify any assumptions they are making. As assumptions are identified, every effort should be made to address or eliminate them. Not all assumptions will be able to be addressed, so the assumptions that are not addressed, should be explicitly included in any discussion of the experiment. It is important to realize the effect the assumptions have on the internal validity, external validity, and construct validity of a study. While the discussion in this paper focused on a classroom study, there is no reason that the general conclusions cannot apply to studies in an industrial setting. Some of the specific assumptions and constructs might change, but there will still be a relationship between assumptions and threats to validity.

Many of the assumptions we made in this example turned out to affect construct validity. For experimental software engineering as a whole, it is important to pay attention to this class of validity criteria. Researchers must understand the theoretical constructs that they are operationalizing in their studies and seek to create comparable representations. Researchers in the social sciences have dealt with the problem of experimenting with unobservable constructs such as experience in other domains. We, as software engineering researchers, should take advantage of their work to improve our experiments.

As we move to a deeper level with our studies and begin to focus on the people and the products as well as the processes, we have to consider more issues. We need to develop and use pretests and posttests to evaluate both the experience of the subjects and the effectiveness of the training. We also need to mature the method for gathering feedback about the set of defects used in our studies.

We realize that it is not feasible to think that all possible assumptions can be eliminated from every software engineering experiment. But, we currently have

a very cursory understanding of these assumptions and their consequences. Even if an assumption cannot be eliminated, it is still important to identify and discuss the assumption. This discussion will allow other researchers to more accurately interpret our results. Furthermore, researchers may be encouraged to replicate studies in which they believe they can address an assumption that identified but not addressed in the original study

We intend to focus our next experiments on the products. In doing so, we will continue to mature in our use of pretests and posttests and we will continue to develop our method of gathering feedback from the subjects on the reality of our seeded defects.

7. Acknowledgements

This work has been partially supported by the NSF-CNPq Readers' Project (CCR-9900307). We would like to thank the reviewers for their insightful comments that have helped to improve this paper. We would like to thank the members of the CMSC 735 class in Fall 2002 semester at the University of Maryland for participating in the study.

8. References

[1] Basili, V. and Reiter, R. "An Investigation of Human Factors in Software Development," *IEEE Computer,* 1979.

[2] Basili, V., Green, S., Laitenberger, O., Shull, F., Sivert, S., and Zelkowitz, M. "The Empirical Investigation of Perspective-based Reading." *Empirical Software Engineering: An International Journal*, 1(2): 133-164.

[3] Basili, V., Shull, F. and Lanubile, F. "Building Knowledge Through a Family of Experiments." *IEEE Transactions on Software Engineering*, 25(4): 456-473. 1999.

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

[5] Carver, J., Shull, F., and Basili, V. "Observational Studies to Accelerate Process Experience in Classroom Studies: An Evaluation." In *Proceedings of 2003 ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 03)*.

[6] Carver, J. "The Impact of Background and Experience on Software Inspections." PhD Thesis, University of Maryland, April 2003. (Also, University of Maryland, Department of Computer Science Technical Report CS-TR-4446).

[7] Carver, J., Van Voorhis, J., and Basili, V. "Investigating the Interaction Between Inspection Process Specificity and Software Development Experience." University of Maryland, Department of Computer Science, Technical Report CS-TR-4532.

[8] Cook, T. and Campbell, D. *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Boston: Houghton Mifflin Co., 1979.

[9] Judd, C., Smith, E., and Kidder, L. *Research Methods in Social Relations*, sixth ed. Harcourt Brace Jovanovich. 1991.

[10] Messick, S. (1995). Validity of psychological assessment: Validation of inferences from persons' responses and performances as scientific inquiry into score meaning. American Psychologist, 50, 741-749.

[11] Parnas, D. and Weiss, D. "Active Design Reviews: Principles and Practice." *Proceedings of 8th International Conference on Software Engineering*, 1985. p. 132-136.

[12] Porter, A., and Johnson, P. "Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies." *IEEE Transactions on Software Engineering*, 23(3): 129-145, 1997.

[13] Powers, D. E. "Self-assessment of Reasoning Skills." Educational Testing Service. Research Report RR-02-22. 2002.

[14] Sauer, C, Jeffery, R., Land, L., and Yetton, P. "The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research." *IEEE Transactions on Software Engineering*, 26(1): 1-14. 2000.

[15] Selby, R., Basili, V., and Baker, T. "Cleanroom Software Development: An Empirical Evaluation." *IEEE Transactions on Software Engineering*, 13(9): 1027-1037

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

[17] Trochim, W. *The Research Methods Knowledge Base, 2nd Edition*. Internet WWW page, at URL: trochim.human.cornell.edu/kb/index.html (version current as of August 02, 2000).

[18] Weinberg, G. M. (1998). The Psychology of Computer Programming Silver Anniversary Edition. New York, Dorset House Publishing.

[19] Zelkowitz, M. and Wallace, D. "Experimental Models for Validating Computer Technology." *IEEE Computer*, 31 (5): 23-31. 1998.