MODELS OF SOFTWARE EXPERIMENTATION

Marvin V. Zelkowitz
Department of Computer Science
University of Maryland
College Park, MD 20742
and NIST/CSL
Gaithersburg, MD 20899

Dolores Wallace
NIST/CSL
Gaithersburg, MD 20899

1. Introduction

In developing a science of software engineering it is important to develop a quantitative component of the field for validating claims in order to replace the folklore that is so prevalent today. However, when one thinks of experimentation, one often has the model of the chemistry laboratory where some chemical (i.e., the independent variable) is altered and the effects of that alteration are measured. Replicating the experiment multiple times allows for statistical techniques to be applied in order to validate the various claims made on the experiment.

This model works well for relatively small experiments, e.g., those that can be accomplished by one or two professionals over a few days. However, many processes in software engineering may require a large team to work several years to complete. The time and costs of such experiments do not allow for easy replication of these phenomena.

Because of this, other models of experimentation have been developed. The case study is a typical example, where a given project is studied in depth in order to determine which characteristics of the project can be generalized to other similar projects. Other models have been used.

We have looked at these models and have developed a classification of 12 different methods of experimentation. In order to test our classification scheme, we have reviewed approximately 700 papers that have been published in the software engineering literature to determine how well this classification model addresses existing software engineering research. We also comment on a related study performed by Tichy [1]. Our results are not as negative as those found by Tichy, but the basic result is the same – software engineering still has a long way to go to improve the quantitative component of the field.

2. Experimentation Classification

We classify the various models for data collection into three broad categories:

1. A *controlled* method provides for multiple instances of an observation in order to provide for statistical validity of the results. This is the classical method of experimental design in fields such as chemistry.

2. An *observational* method will collect relevant data as a project develops. In general, there is relatively little control over the development process other than using the new technology that is being studied. Astronomy is a discipline that often uses such techniques.

3. An *historical* method collects data from projects that have already been completed. The data already exists; it is only necessary to analyze what is already there. This is most like a field such as archaeology.

From these three broad categories, we subdivide them as follows:

I. Controlled methods

1. Replicated. Several projects are observed as they develop in order to determine the effects of the independent variable. Due to the high costs of such experiments, they are extremely rare.

2. Synthetic environments. These represent replicated experiments, but generally developed in an artificial setting, e.g., often in a university. For example, taking an artifact out of context, e.g., a design document for a single component, and then developing source code for it, would probably represent such a synthetic experiment.

3. Dynamic analysis. The project is replicated using real project data. This is most common in testing products using real project data, e.g., a code complexity tool that processes the modules of a large development.

4. Simulation. This is similar to the dynamic analysis method, except that artificial, but realistic, data is used, such as assuming that all data adheres to a normal distribution of values.

II. Observational methods

5. Project monitoring. Collect data on a project with no preconceived notion of exactly what is to be collected (e.g., a simple accounting routine connected to all runs of a program).

6. Case study. Specific data collected as a project develops by individuals who are part of the development group. Specific changes to the development organization may be part of the environment in order to evaluate such changes.

7. Assertion. A relatively simple form of case study that does not meet rigorous scientific standards of experimentation. We added this classification since it is so prevalent in software engineering. A new technology is developed, and then a simple experiment is used to justify the technology. In essence it says, "We tried it and we like it."

8. Survey. An outside group comes to observe a development and collect data on its development. It is related to the case study above, but the experimenters have less control over modifying the development practices as does the case study group.

III. Historical methods

9. Literature search. The experimenter views previously published papers in order to arrive at a conclusion.

10. Legacy data. Data from a completed project is studied in order to determine if certain experimental conditions were met.

11. Lessons-learned data. Interviews with project personnel and a study of project documentation from a completed project can be used to determine qualitative results from certain technologies. This is related to the Legacy data method, above; however, it has less quantitative data to work with.

12. Static analysis. Artifacts of a completed project (e.g., source code, design documents, test results) are processed in order to determine characteristics of a completed project.

3. Model Validation

A preliminary check of these classifications was accomplished by both of us by reviewing the papers in the ICSE 15 (1993) and ICSE 18 (1996) proceedings. We independently classified each paper, and compared our results. We agreed on 90%-95% of the papers and resolved all our differences.

In order to be able to classify every paper, we had to add two additional classifications:

13. Not applicable. The published paper does not discuss a new technology, e.g., a survey paper on existing approaches.

14. No experimentation. The paper presents a new technology, but makes no claims as to experimental validity. Note that "No experimentation" is not synonymous with "Bad experimentation." Some papers are simply not appropriate for experimentation, e.g., a paper describing a new formal theory often has no experimental component; however, you would expect that later papers on that subject would include such experimentation.

The results of our initial study can be summarized by the following table:

Method	Count	%*	Count	%*
	ICSE15 -1993		ICSE18 - 1996	
Not applicable	1	–	6	–
No experimentation	16	34.0	20	37.7
Replicated	0	0	2	3.8
Synthetic	1	2.1	1	1.9
Dynamic analysis	0	0	2	3.8
Simulation	1	2.1	1	1.9
Project monitoring	0	0	0	0
Case study	8	17.0	9	17.0
Assertion	6	12.8	4	7.5
Survey	0	0	1	1.9
Literature search	3	6.4	3	5.7
Legacy data	5	10.6	5	9.4
Lessons learned	4	8.5	5	9.4
Static analysis	3	6.4	0	0

* - Percentages do not include "Not applicable" category.

Every paper was relatively easy to classify into one of the 14 given categories, although with our relatively small sample, not every possible class was present.

While Tichy [1] found about 50% of the papers as containing no experimental data, our finer granularity shows that we found only around 1/3 of the papers with no experimental data, still a very high number. However, when you also include the Assertion category as a weak form of validation, then we get a figure of approximately 45%, which is much closer to Tichy's figure.

At the present time we are expanding our literature search by classifying a larger set of papers. Aided by Dale Walters of NIST, we are looking at approximately 700 software engineering papers published in 1985, 1990, and 1995 in order to see: (1) if our classification model works for a larger class of papers, and (2) if there have been any trends in the type of experimental data collected over the last 10 years. One would hope that the experimental paradigm is more prevalent today, and we expect to report on the full survey in the near future.

References

[1] Tichy W. F., P. Lukowicz, L. Prechelt, and E. A. Heinz, Experimental evaluation in computer science: A quantitative study, *J. of Systems and Software* Vol. 28, No. 1 (1995) 9-18.