passed $15.2-billion for 1994. The problem is most rampant in Indonesia and Kuwait, where about 99% of all software is copied illegally. [*Toronto Financial Post,* 6 May 1995, p. 8, from Edupage, 7 May 1995.]

British man convicted as malicious virus writer (George Smith)

Chris Pile (Black Baron), a 26-year-old unemployed computer programmer, pleaded guilty to 11 charges of computer viruses activities (including SMEG/Pathogen and SMEG/Queeg). This was the first conviction under the British Computer Misuse Act of 1990. (R 17 16) has more details. PGN]

ONE-LINER SUMMARIES of additional items from the on-line Risks Forum:

- Ford Motor Co promotional floppy disk contains monkey virus (R 17 23)
- Randal Schwartz convicted after finding security flaws in Intel (R 17 23,28)
- German intruder forges White House messages (R 17 31,32)
- Sony satellite dishes remotely reprogrammable? (R 17 33)
- Microsoft Network E-mail binaries can contain executables (R 17 31,32,33)
- Phony ATM installed on High St in London, nets £120K (R 17 34)
- Emergency call-boxes ripped off, cell-phone serial nos. reused (R 17 35)
- German telephone card system cracked, many free calls made (R 17 36)
- British Telecom replaces payphone software after flaw exploited (R 17 36)
- London Underground gets hacked by insider posting nasty messages (R 17 36)
- Cardiff software shipped Teleforms 4.0 with self-destruct timebomb (R 17 36)

Experimental Design and Analysis in Software Engineering Part 5: Analyzing the Data

Shari Lawrence Pfleeger
Centre for Software Reliability
City University
Northampton Square
London EC1V 0HB England
phone: +44-71-477-8426 – fax: +44-71-477-8585
shari@csr.city.ac.uk

After you have run an experiment and collected the relevant data, you must analyze it in an appropriate way. This article describes the items you must consider in choosing the analysis techniques. We describe typical situations for which you may be performing an experiment, and what technique is most appropriate for each situation. Specific statistical techniques are described and used in the discussion, but the details of each statistical approach (including formulae and references to other statistical textbooks) are found in [Kitchenham 1992], [Caulcutt 1993] and [Chatfield 1993].

There are three major items to consider when choosing your analysis techniques: the nature of the data you collected, why you performed the experiment, and the type of experimental design you used. We consider each of these in turn.

DISTRIBUTION OF THE DATA

The nature of your data will help you to decide what analysis techniques are available to you. Thus, it is very important for you to understand the data as a sample from a larger population of all the data you could have gathered, given infinite resources. Because you do not have infinite resources, you are using your relatively small sample to generalize to that larger population, so the characteristics of the population are important. Many statistical techniques assume that the data is normally distributed, and your sample is randomly chosen from that larger distribution.

However, most software-related measurements are not normally distributed. Whereas a normal distribution is continuous and symmetric about its mean, much software data is discrete and not symmetric. In fact, it is unusual for software data to be a random sample from a well-defined population. For example, the distribution of software module size data can be very different from a normal distribution. You can tell if your data is normal by doing several tests. For instance, in a normal distribution, the three major indicators of central tendency (mean, median and mode) are the same. [Kitchenham 1992] describes several others. If you are not sure whether your data is normal or not, you must assume that it is not, and use techniques for evaluating non- normal data.

There are several ways to analyze data that is non-normal:

- Use robust statistics and non-parametric analysis methods.

- Transform your measurements to a scale (such as the logarithmic scale) that conforms more closely to a normal distribution.
- Determine the true underlying distribution and choose techniques appropriate to it.

In the examples that follow, we will consider both normal and non-normal cases, depending on the type of data.

PURPOSE OF THE EXPERIMENT

In Part 2, we noted four major reasons to conduct a formal experiment:

- to confirm a theory
- to explore a relationship
- to evaluate the accuracy of a model
- to validate a measure

Each of these requires analysis carefully designed to meet the stated experimental objective. In particular, the objective is expressed formally in terms of the hypothesis, and the analysis must address the hypothesis directly.

Confirming a theory

Your experiment may be designed to explore the truth of a theory. The theory usually states that use of a certain method, tool or technique (the treatment) has a particular effect on the experimental subjects, making it better in some way than another treatment (usually the existing method, tool or technique). For example, you may want to investigate the effect of the cleanroom technique by comparing it with your existing testing methods. The usual analysis approach for this situation is *analysis of variance*. That is, you consider two populations, the one that uses the old technique and the one that uses the new, and you do a statistical test to see if the difference in treatment results is statistically significant.

There are two cases to consider: normal data and non-normal data. If the data comes from a normal distribution and you are comparing two groups, you can use the *Student's t-test* to analyze the effects of the two treatments. If you have more than two groups to compare, a more general analysis of variance test, using the *F statistic*, is appropriate. Both of these are described in most statistics books.

For example, suppose you are investigating the effect on productivity of the use of a new tool. You have two groups that are otherwise equal except for use of the tool: group A is using the existing method, while group B is using the tool to perform the designated task. You are measuring productivity in terms of thousands of delivered source code instructions per month, and you feel confident that the productivity data comes from a normal distribution. You can use a Student's t-test to compare group A's productivity data with group B's to see if the use of the tool has made a significant change in productivity.

On the other hand, suppose you want to investigate whether the cleanroom technique yields higher-quality code than your current testing technique. Your hypothesis is stated as

> Code developed using the cleanroom technique has the same number of defects per lines of code as code developed using current testing techniques.

You collect data on number of defects per lines of code for each of two groups, and you seek an analysis technique that will tell you whether or not the data supports the hypothesis. Here, the data on defects per lines of code is not normally distributed. You can analyze the defect data by ranking it and using the Kruskal-Wallis test to tell you if the mean rank of the cleanroom defects is lower than that of the non-cleanroom data.

Exploring a relationship

Often, an experiment is designed to determine the relationship among data points describing one variable or across multiple variables. For example, you may be interested in knowing the normal ranges of productivity or quality on your projects, so that you have a baseline to compare for the future. A case study may be more appropriate for this objective, but you may want to answer this question as part of a larger experiment. There are three techniques to use to answer questions about a relationship: box plots, scatter diagrams, and correlation analysis.

A *box plot* can depict for you a summary of the range of a set of data. It shows you where most of the data is clustered and where any outlier data may be. Whereas a box plot shows information about one variable, a *scatter diagram* depicts the relationship between two variables. By viewing the relative positions of pairs of data points, you can visually determine the likelihood of an underlying relationship between the variables. You can also identify data points that are atypical, because they are not organized or clustered in the same way as the other data points.

Correlation analysis goes a step further than a scatter diagram by using statistical methods to confirm whether there is a true relationship between two attributes. Correlation analysis can be done in two ways: by generating measures of association that indicate the closeness of the behavior of the two variables, or by generating an equation that describes that behavior. For example, you may be investigating the relationship between number of control paths in a module and the number of defects identified in the module. A statistical analysis of these two variables can yield a measure of association, between -1 and 1, that indicates whether a high number of control paths usually means a large number of defects. That is, the measure of association is 1 if there is a perfect positive linear relationship between the two variables, -1 if there is a perfect negative linear relationship, and 0 if there is no relationship. If you want to be able to predict the number of defects from the number of control paths, then the measure of association is insufficient; correlation analysis techniques can generate an equation of the form:

$$(\text{number of defects}) = (\text{constant1})(\text{number of control paths}) + (\text{constant2})$$

to permit such prediction.

When measures of association are sufficient, it is important to know if the data is normally distributed or not. For normally-distributed values, a *Pearson correlation coefficient* is a measure of association that indicates whether the two variables are highly correlated or not. For non-normal data, you must rank the data and use the *Spearman rank correlation coefficient* as a measure of association. An alternative for non-normal data is the *Kendall robust correlation coefficient*, which investigates the relationships among pairs of data points and can identify partial correlations. If the ranking contains a large number of tied values (that is, repeats of the same value, so that they rank the same), a Λ^2 *test on a contingency* table can be used to test the association between the variables.

When you need to understand the nature of the association as well, you can use *linear regression* to generate an equation to describe the relationship between two variables you are examining. This technique produces a line that minimizes the sum of the squares of the residuals. That is, linear regression minimizes the distance from the line to the points off the line. For more than two variables, *multivariate regression* can be used.

The data to which regression is applied does not need to be normally distributed. However, regression techniques assume that the residuals (the distance from each point to the line) are normally distributed. When they are not, two techniques can be used instead of standard regression. *Theil's robust regression* uses the slopes of a carefully-defined set of lines to determine the slope of the regression line. Or, the data points can be transformed to a more normal distribution by viewing their *logarithms*, rather than the data itself; then, regression is applied to the new data to generate a formula, and conversion back to non-logarithmic variables completes the process.

Evaluating the accuracy of a model

For many software engineering tasks, a model of behavior is used to predict what should happen. These predictions aid the project manager in making major decisions about resource planning and the duration or extent of activities. For example, the manager may model the likely cost of a project as a basis for decisions about resource allocation. Alternatively, he or she may use a defect model to predict how long to test before turning over the product to the customer. It is important to be able to evaluate the accuracy of these models. In these cases, although the intent of the experiment is different from confirming a theory or exploring a relationship, the analysis techniques are the same. The model in question generates a set of predicted data, and it is to be compared with the set of actual data. Thus, the steps described for confirming a theory can be followed here as well. If the data is normally distributed, a Student's t-test can be used to determine if there is a significant difference between the predictions and the actual numbers. If the data is not normally distributed, then the Kruskal-Wallis test will suffice.

Validating a measure

Validating a measure means verifying that the measure actually captures the attribute it claims to reflect. For example, the McCabe cyclomatic number [McCabe 1976] claims to capture the complexity of source code, when in fact it actually measures the number of decision points (plus one) in the code. Experiments are often designed to validate a measure by exploring the relationship between the measure and data that is known to be highly correlated with the attribute in question. For this reason, the analysis techniques used to explore a relationship are also the appropriate ones for validating a measure.

DESIGN CONSIDERATIONS

The experimental design must be considered in choosing the analysis techniques. At the same time, the complexity of analysis can influence the design chosen, as noted in Part 3. Multiple groups usually generate the need to use the F statistic with a full-blown analysis of variance, rather than a simple Student t-test with two groups. For complex factorial designs with more than two factors, more sophisticated tests of association and significance must be used. Statistical techniques can be used to account for the effects of one set of variables on others, or to compensate for timing or learning effects. These techniques are beyond the scope of this series of articles, and you should consult a statistician for help with this type of design.

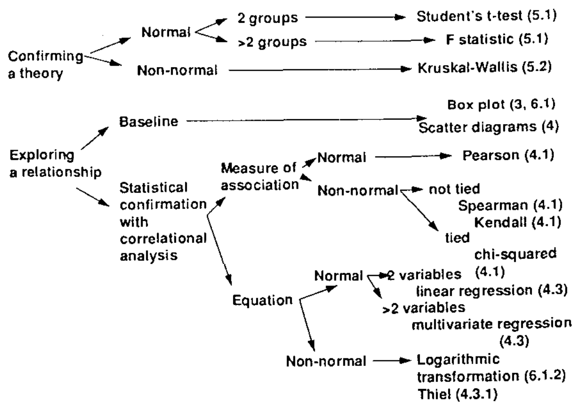

Table 2. Decision tree for analysis techniques

DECISION TREE

To help you understand when to use one analysis technique or another, you can use the decision tree in Table 2 to take into account the major considerations discussed in this section. The decision tree is to be read from left to right. Beginning with the objective of your experiment, move along the branch that fits your situation until you reach a leaf node with the appropriate analysis technique(s). In parentheses after each analysis technique is a cross-reference to [Kitchenham 1992]; this reference will enable you to read about that technique in more detail, complete with examples. For simplicity, the number of experimental objectives had been reduced from four to two, consistent with the discussion above.

SUMMARY

We have described the key activities necessary for designing and analyzing an experiment in software engineering. We began by explaining how to choose an appropriate research technique to fit the goals of your project. In particular, we taught you how to state your hypothesis and determine how much control you need over the variables involved. If control is not possible, then a formal experiment is not possible; a case study may be a better approach.

Next, we explained the principles of formal experimentation. We listed the six stages of an experiment: conception, design, preparation, execution, analysis and dissemination. The design of an experiment was discussed in some detail. In particular, we pointed out that you must consider the need for replication, randomization and local control in any experiment that you plan to perform. We showed you how you can think of your design in terms of two types of relationships between factors (crossing and nesting), and we described several issues to be considered when selecting an appropriate design.

Once your experimental design was determined, we discussed how to analyze the results. We explained how the distribution of the data can influence the choice of analysis technique, as can the purpose of the experiment and the design considerations.

We hope these articles are useful not only in helping you set up your own experiments, but also in assessing the work of others. There is a profusion of experiments reported in the software engineering literature, many of which suggest that you adopt a particular method, tool or technique. With the analysis suggested here, you should develop a critical eye that will enable you to determine when the reported results are valid and whether the results can be applied to your particular situation or organization.

As I am in the process of moving back to the US, I have asked Barbara Kitchenham of the UK National Computing Centre to take over for the next few installments. She will write a series of articles addressing case studies, much in the same way that we have explored the major issues involved in conducting experiments. Both the experimental and case study work was performed as part of the DESMET project, led by the National Computing Centre and funded by the UK Department of Trade and Industry.

REFERENCES

Caulcutt, Roland (1991), *Statistics in Research and Development*, Chapman and Hall, London, England.

Chatfield, Christopher (1993), *Statistics for Technology*, Chapman and Hall, London, England.

Kitchenham, Barbara (1992), *DESMET Handbook of Data Collection and Metrication Book 3: Analysing Software Data*, National Computer Centre, Manchester, England.

REUSE EMPHASIZED AT NEXT PROCESS WORKSHOP

Barry Boehm
USC Computer Science Dept.
Los Angeles, CA 90089-0781, USA
boehm@sunset.usc.edu

The 10th International Software Process Workshop, being held June 17-19, 1996, in Dijon, France, will be emphasizing a software reuse-oriented theme: Process Support of Software Product Lines.

Much of the technology currently available to support the software process has focused on the process of developing and evolving a single software product. Increasingly, organizations are finding advantages in product-line software approaches, involving investments in domain engineering, product line architectures, and rapid applications composition with extensive use of commercial-off-the-shelf (COTS) and other reusable software assets. These new approaches involve significant software process challenges: for example, large-scale software packages (COTS and in-house) often provide so much of an application system's desired functionality that the most effective software approach is for the COTS/reuse capabilities to drive the requirements, rather than the traditional requirements-to-capabilities process model.

Candidate issues to be addressed by the Workshop are Reuse-Sensitive Process Models; Product-Line Oriented Process Models; Reuse of Process Elements; and Effect of Product-Line Considerations on Current Process Technology. Examples of more detailed issues are: How are reuse considerations reflected in models of requirements engineering, architecting, risk management, development, testing, evolution, and process maturity? How are process models affected if the focus is not on a single applications product, but on a product line family of applications? How do these considerations of process support of software product lines affect process representation languages, process enactment support, process management tools, the role of human beings in the software process, and the selection of representative example problems for the software process community to address?

The workshop will consist of intensive discussions of these issues by at most 35 participants, selected on a basis of submitted position papers. Prospective participants should submit a maximum three-page position paper by 5 January 1996, explicitly addressing the workshop theme, and suitable for publication in the proceedings. Papers (7 copies or email in ascii or Postscript format only) should be sent to the address above. A Web page at http://sunset.usc.edu/Events.html provides more information on the Workshop.