Preliminary Guidelines for Empirical Research in Software Engineering

Barbara A. Kitchenham, *Member*, *IEEE Computer Society*,
Shari Lawrence Pfleeger, *Member*, *IEEE*, Lesley M. Pickard, Peter W. Jones,
David C. Hoaglin, Khaled El Emam, *Member*, *IEEE Computer Society*, and
Jarrett Rosenberg, *Member*, *IEEE Computer Society*

Abstract—Empirical software engineering research needs research guidelines to improve the research and reporting processes. We propose a preliminary set of research guidelines aimed at stimulating discussion among software researchers. They are based on a review of research guidelines developed for medical researchers and on our own experience in doing and reviewing software engineering research. The guidelines are intended to assist researchers, reviewers, and meta-analysts in designing, conducting, and evaluating empirical studies. Editorial boards of software engineering journals may wish to use our recommendations as a basis for developing guidelines for reviewers and for framing policies for dealing with the design, data collection, and analysis and reporting of empirical studies.

Index Terms—Empirical software research, research guidelines, statistical mistakes.

1 Introduction

We have spent many years both undertaking empirical studies in software engineering ourselves and reviewing reports of empirical studies submitted to journals or presented as postgraduate theses or dissertations. In our view, the standard of empirical software engineering research is poor. This includes case studies, surveys, and formal experiments, whether observed in the field or in a laboratory or classroom. This statement is not a criticism of software researchers in particular; many applied disciplines have problems performing empirical studies. For example, Yancey [50] found many articles in the *American Journal of Surgery* (1987 and 1988) with "methodologic errors so serious as to render invalid the conclusions of the authors." McGuigan [31] reviewed 164 papers that included numerical results that were published in the *British Journal of Psychiatry* in 1993 and found that 40 percent of them had statistical errors. When Welch and Gabbe [48] reviewed clinical articles in six issues of the *American Journal of Obstetrics*, they found more than half the studies impossible to assess because the statistical techniques used were not reported in sufficient detail. Furthermore, nearly one third of the articles contained inappropriate uses of statistics. If researchers have difficulty in a discipline such as medicine, which has a rich history of empirical research, it is hardly surprising that software engineering researchers have problems.

In a previous investigation of the use of meta-analysis in software engineering [34], three of us identified the need to assess the quality of the individual studies included in a meta-analysis. In this paper, we extend those ideas to discuss several guidelines that can be used both to improve the quality of on-going and proposed empirical studies and to encourage critical assessment of existing studies. We believe that adoption of such guidelines will not only improve the quality of individual studies but will also increase the likelihood that we can use meta-analysis to combine the results of related studies. The guidelines presented in this paper are a first attempt to formulate a set of guidelines. There needs to be a wider debate before the software engineering research community can develop and agree on definitive guidelines.

Before we describe our guidelines, it may be helpful to you to understand who we are and how we developed these guidelines. Kitchenham, Pickard, Pfleeger, and El-Emam are software engineering researchers with backgrounds in statistics as well as computer science. We regularly review papers and dissertations, and we often participate in empirical research. Rosenberg is a statistician who applies statistical methods to software engineering problems. Jones is a medical statistician with experience in developing standards for improving medical research studies. Hoaglin is a statistician who has long been interested in software and computing. He reviewed eight papers published in *Transactions on Software Engineering* in the last few years. These papers were not chosen at random.

- *B.A. Kitchenham and L.M. Pickard are with the Computer Science Department, Keele University, Keele, Staffordshire ST5 5BG, UK. E-mail: {barbara, lesley}@cs.keele.ac.uk.*
- *S.L. Pfleeger is with the RAND Corporation, 1200 South Hayes St., Arlington, VA 22202-5050. E-mail: shari_pfleeger@rand.org.*
- *P.W. Jones is with the Mathematics Department, Keele University, Keele, Staffordshire ST5 5BG, UK. E-mail: maa11@maths.keele.ac.uk.*
- *D.C. Hoaglin is with Abt Associates Inc., 55 Wheeler Street, Cambridge, MA 02138. E-mail: Dave_Hoaglin@abtassoc.com.*
- *K. El Emam is with the National Research Council of Canada Institute for Information Technology, Montreal Road, Building M-50 Ottowa, Ontario, K1J 8H5 Canada. E-mail: khaled.el-emam@seg-nrc.org.*
- *J. Rosenberg is with Sun Microsystems, San Antonio Road, MS MPK17-307 Palo Alto, CA 94303. E-mail: Rosenberg@eng.sun.com.*

Manuscript received 30 Dec. 1999; revised 26 Jan. 2001; accepted 23 Apr. 2001.
Recommended for acceptance by M. Jazayeri.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 111161.

0098-5589/02/$17.00 © 2002 IEEE

Rather, they were selected (by those of us whose primary focus is software engineering) because their authors are well-known for their empirical software engineering work and because their techniques are typical of papers submitted to this journal. Hoaglin's independent comments on these papers confirmed our suspicions that the current state of empirical studies as published in *Transactions on Software Engineering* is similar to that found in medical studies. He found examples of poor experimental design, inappropriate use of statistical techniques and conclusions that did not follow from the reported results. We omit the titles of these papers. We want the focus of our guidelines to be overall improvement of our discipline, not finger-pointing at previous work. We do, however, cite papers that include specific statistical mistakes when they help illustrate the reason that a particular guideline should be followed.

The main sources for this paper, apart from our own experience, are:

- The Yancey paper already mentioned. Yancey identifies 10 rules for reading clinical research results. Many of the rules can also serve as guidelines for authors.
- A paper by Sacks et al. [43] that considers quality criteria for meta-analyses of randomized controlled trials. Sacks et al. point out that the quality of papers included in a meta-analysis is important. In particular, they suggest considering the quality of features such as the randomization process, the statistical analysis, and the handling of withdrawals.
- A paper on guidelines for contributors to journals by Altman [1].
- The guidelines for statistical review of general papers and clinical trials prepared by the *British Medical Journal*. (These guidelines are listed in Altman et al. [3], chapter 10 of Gardner and Altman [14], and on the journal's web page: http://www.bmj.com/advice.)
- A book by Lang and Secic [28] with guidelines for reporting medical statistics.
- The CONSORT statement on reporting the results of randomized trials in medicine [4]. This statement has been adopted by 70 medical journals.
- A paper defining guidelines for reporting results of statistical analysis in the *Japanese Journal of Clinical Oncology* [12]. This paper also discusses common statistical errors.
- A paper defining guidelines for the American Psychological Society [49].
- Three papers discussing the types of statistical errors found in medical papers [31], [2], [38].

We have concentrated on medical guidelines because medical statisticians have been particularly active in pointing out the poor standards of statistical analysis in their journals. In addition, we also reviewed the guidelines of the American Psychological Association. We have adapted advice from the above sources to the problems of empirical studies of software engineering phenomena. Some problems with statistics arise because there are methodological difficulties applying standard statistical procedures to software experiments. Nonetheless, the majority of problems result from lack of statistical expertise in the empirical research community. Altman suggests the same is true in medicine [2]. He says "The main reason for the plethora of statistical errors is that the majority of statistical analyses are performed by people with an inadequate understanding of statistical methods. They are then peer reviewed by people who are generally no more knowledgeable." Our guidelines consider both types of problem, but some methodological problems are, as yet, unsolved. In these cases, we point out the problems, but we have no ready solutions.

In principle, empirical guidelines can represent the concerns of many different parties

- The reader of a published paper.
- The reviewer of a paper prior to its publication.
- The authors of a paper.
- Researchers planning an empirical study.
- A meta-analyst wanting to combine information from different studies of the same phenomenon.
- A journal editorial board.

In addition, empirical guidelines are often specialized to consider particular types of study e.g., randomized trials, surveys, exploratory studies. Clearly, the particular requirements for a set of guidelines influence their content and format. In the long term, if the software community accepts the need for experimental guidelines, we would expect to find specialized guidelines for different purposes. In this paper, however, we are concerned with developing guidelines to assist researchers to avoid major pitfalls in their research activities and to report their research correctly. Some guidelines pertain to particular types of studies but most are fairly general. We believe that our guidelines will cover most of the issues of relevance to the researchers. We cannot claim that the guidelines are necessarily compatible with all the requirements of other interested parties.

We consider guidelines for what do to and what not to do under six basic topic areas:

- Experimental context,
- Experimental design,
- Conduct of the experiment and data collection
- Analysis,
- Presentation of results, and
- Interpretation of results.

The experimental guidelines and advice we reviewed were at varying levels of abstraction. Some appeared to be metaguidelines identifying the goal of a set of guidelines. For example, the BMJ General Guidelines say "Select a study design that is appropriate to achieve the study objective" which can be considered a metagoal for all the design level guidelines. Alternatively, some guidelines are very detailed. For example, Lang and Secic [28] say "When reporting percentages, always give the numerators and denominators of the calculations" which is more like a checklist item for a more general guideline "Provide appropriate descriptive statistics." We have tried to ensure that our guidelines are at a level below metaguidelines, which are incorporated into the introduction to each set of guidelines, and above the very detailed level guidelines, which are incorporated as checklists associated with the relevant guideline.

2 Experimental Context

2.1 Introduction

Most medical guidelines have little to say about experimental context. However, we regard experimental context as extremely important for software engineering research. Experimental context has three elements:

1. Background information about the industrial circumstances in which an empirical study takes place or in which a new software engineering technique is developed.
2. Discussion of the research hypotheses and how they were derived.
3. Information about related research.

The main goals of context guidelines are:

1. To ensure that the objectives of the research are properly defined
2. To ensure that the description of the research provides enough detail for other researchers and for practitioners.

2.2 Context Guidelines

C1: *Be sure to specify as much of the industrial context as possible. In particular, clearly define the entities, attributes, and measures that are capturing the contextual information.*

Industrial context information is important in two types of empirical software engineering studies: observational studies (i.e., in situ studies of industrial practice) and formal experiments evaluating techniques developed in industry (e.g., inspections or design patterns). We discuss these situations below.

2.2.1 Observational Studies

There is an immense variety to be found in development procedures, organizational culture, and products. This breadth implies that empirical studies based on observing or measuring some aspect of software development in a particular company must report a great deal of contextual information if any results and their implications are to be properly understood. Researchers need to identify the particular factors that might affect the generality and utility of the conclusions. For example, they may need to identify factors such as:

1. The industry in which products are used (e.g., banking, consumer goods, telecommunications, travel).
2. The nature of the software development organization (e.g., in-house information systems department, independent software supplier).
3. The skills and experience of software staff (e.g., with a language, a method, a tool, an application domain)
4. The type of software products used (e.g., a design tool, a compiler)
5. The software processes being used (e.g., a company-standard process, the quality assurance procedures, the configuration management process).

Such information is essential if the same or other researchers want to replicate a study or include it in a meta-analysis. Further, if practitioners want to use experimental results, they need to be sure that the results are applicable in their specific circumstances

Unlike other disciplines, software engineering has no well-defined standards for determining what contextual information should be included in the study design, collected during the study, and reported with the results. Standard contextual information allows us to compare and contrast similar studies, and to build a fabric of understanding from individual threads of evidence about a tool, technique or practice. Research guidelines are incomplete, because identification of standards for reporting research context requires further research in two areas:

1. Identification of general confounding factors for specific types of empirical studies. For example, El-Emam et al. [10] have demonstrated that size is a confounding factor when using object-oriented coupling measures to predict fault-proneness, but more research is needed to identify other confounding factors for other standard types of experiments and analyses.
2. Specification of a taxonomy or an ontology of context. Kitchenham et al. [23] have suggested an ontology of software maintenance aimed at defining factors that need to be reported in empirical studies. This work identifies a very large number of factors but does not offer any advice as to which factors are most important, nor what minimum reporting requirements should be.

It is important not only to identify appropriate contextual variables but also to measure them consistently. Again, software engineering studies suffer from specific problems because software measurements are not standardized. Fenton and Pfleeger [11] provide many examples of product, process, and resource characteristics that are measured inconsistently across different development and maintenance organizations. For example, there are multiple standards for counting function-points, and confusion exists about what is meant by counting a line of code. Similarly, several measures purport to measure "complexity," but it is not clear what aspect of complexity is being measured; neither is it clear whether that aspect is being measured appropriately or consistently. Kitchenham et al. [24] suggest several ways to help researchers and practitioners focus on what they mean by entities and attributes and how to define and measure them more consistently. Still, much work remains in making sure we mean the same things when we discuss and measure variables in empirical studies.

2.2.2 Formal Experiments

For formal experiments evaluating techniques developed in industry (e.g., inspections, or design patterns), we need to be sure that the versions of the techniques that we evaluate and the outcome measures we use for testing are not oversimplified. Unlike techniques developed in academia, industrial techniques have usually been developed in a rich complex setting. We need to understand how the technique works in an industrial setting before we develop an abstract version for formal experimentation.

Research on inspections illustrates some of the problems of over-simplification. Many inspection experiments have concentrated on assessing easily quantifiable benefits and ignored other types of benefit. In the context of experiments aimed at optimization, this is a particular problem since optimization in one dimension can suboptimize overall performance. For example, some studies have suggested that inspection meetings are not necessary to maximize defect detection, the hypothesized synergy of inspection meetings does not exist [20], [27], [47]. This lack of effect may be true, but it does not imply that inspection meetings can be abandoned. We can omit the inspection meetings only when *all* other benefits are shown to be illusionary or unaffected by the meeting process.

Observational studies give a better overview of the full range of benefits accruing from inspections. For example, a study by Doolan [9] identified a number of benefits of inspections other than simple fault detection:

1. Inspections promote teamwork and are a good means of developing team spirit in a large group.
2. Inspections are an excellent means for transferring technology, especially if key people leave the project.
3. Inspections provide on-the-job training on standards, technical material, organizational culture, and the inspection process itself.
4. Inspectors can identify root causes of defects.

Siy and Votta [44] conclude that inspections improve the maintainability of code by identifying "soft issues" such as making code conform to standards, minimizing redundancies, improving safety and portability, and raising the quality of the documentation. Each of these benefits makes the code easier to understand and change. However, absence of these characteristics would not necessarily be classified as a defect because each one does not always lead to a visible, particular failure. Thus, formal experiments based only on defect counts provide an incomplete picture of the true costs and benefits.

C2: *If a specific hypothesis is being tested, state it clearly prior to performing the study and discuss the theory from which it is derived, so that its implications are apparent.*

Confirmatory experiments are the foundation of experimental science. They are designed to test formal hypotheses as rigorously as possible. In particular, hypotheses are formulated during experimental design prior to the collection or analysis of any data.

Software engineering researchers are becoming more familiar with the concept of stating their scientific hypotheses. However, all too often the so-called hypothesis is simply a statement of the tests to be performed. For example, if researchers want to know whether there is a relationship between the cyclomatic number and the number of faults found in a module, they may state the null hypothesis as "there is no correlation between cyclomatic number and faults found." We call this statement a shallow hypothesis because it does not reflect an underlying, explanatory theory. That is, whatever the result of the experiment, we will not be increasing our software engineering knowledge in any significant way. Furthermore, this approach is ultimately sterile since it leads to experimental results that are not of great interest.

Unfortunately, the inspection and reading literature offers many examples of shallow hypotheses and unexciting research. Often, the hypothesis under test is artificial, in the sense that it is aimed at optimization within a specific context, rather than testing an aspect of a theory. That is, there is no well-defined theory about how software engineers introduce defects into software artifacts, nor any theory about how they recognize those defects. Even more important, there is no real theory about why inspections work, even though there is evidence that they do. Useful research may flow solely from empirical observations. For example, the development of the smallpox vaccination by Edward Jenner was based solely on the observation that women who had had cowpox did not catch smallpox [15]. However, Jenner succeeded because there was an underlying cause-effect relationship, even though current medical research had not yet discovered it. Unfortunately, empirical studies of software engineering phenomena are often contradictory. Without any underlying theories, we cannot understand the reason why empirical studies are inconsistent. When we do have consistent results, as in the case of inspections, we cannot be sure how to improve or optimize the result if we do not understand why the phenomenon occurs.

Other scientific disciplines search for a deep hypothesis that arises from an underlying theory, so that testing the hypothesis tests the validity of the theory. For example, rather than merely documenting a relationship between cyclomatic number and faults found, we can use theories of cognition and problem-solving to help us understand the effects of complexity on software. Vinter et al. [46] provide a good example of deriving hypothesis from theory in their work, concerned with assessing the effectiveness of formal methods. Vinter et al. investigated the logical errors made by experienced software engineers (who knew their performance was being tested) and compared their results with cognitive psychology studies of reasoning errors found in the general population. They found many similarities in the errors made by software engineers and the general population. The importance of their research is that it considered the claims made for formalisms in terms of the psychological assumptions underpinning those claims. Without the link from theory to hypothesis, empirical results cannot contribute to a wider body of knowledge.

C3: *If the research is exploratory, state clearly and, prior to data analysis, what questions the investigation is intended to address and how it will address them.*

There is always a tension between gaining the maximum value from an experiment by performing many subsidiary analyses to look for areas for future research and having the most convincing experimental results by concentrating on a single purpose confirmatory study. Many researchers will perform a complex study to investigate multiple hypotheses, but they emphasize only the results that bear on the primary hypothesis. The other analyses act as prestudies that allow them to formulate further hypotheses to be tested in subsequent experiments. Such prestudies can also ensure that the researchers have sufficient information to plan future experiments better, by predicting the number of experimental subjects needed and the expected power of the test.

Thus, exploratory studies are an important mechanism for generating hypotheses and guiding further research activities. However, if researchers apply extensive statistical analyses to exploratory studies, the analyses may suffer from methodological problems:

1. Too many tests. Researchers may perform so many tests that a large number of statistically significant results occur by chance.
2. Fishing for results. Researchers may trawl all possible combinations and subsets of the data, however unlikely, in order to find statistically significant results.

We present analysis guidelines later that help to avoid these problems, but the starting point is to define the research questions in advance.

Defining research questions in advance addresses another major problem with exploratory analysis, which is the temptation to undertake too many research activities in a single study. For example, Ropponen and Lyytinen [40] surveyed project managers about their experiences of various risk factors. They performed principal component analysis on the results and interpreted the six largest principal components as identifying components of project risk. In effect, they had generated hypotheses about the nature of risk component from their analysis. They did not report that it was now necessary to perform a second independent survey to test their hypotheses. In fact, they used the principal components in further exploratory correlation studies to identify strategies for managing those components. Furthermore, they attempted to validate the survey instrument using the same data set. They derived Cronbach alpha statistics, which measure the agreement among respondents to questions that address the same topic [7], using the risk components identified by the principal component analysis. They found high values and concluded that they had validated their survey instrument. However, one would expect the Cronbach alpha values to be high because they measure structure known to be present in the data.

C4: *Describe research that is similar to, or has a bearing on, the current research and how current work relates to it.*

The relationship between the current research activity and other research should be defined, so that researchers can combine to build an integrated body of knowledge about software engineering phenomena.

3 Experimental Design

3.1 Introduction

The study design describes the products, resources and processes involved in the study, including:

- the population being studied,
- the rationale and technique for sampling from that population,
- the process for allocating and administering the treatments (the term "intervention" is often used as an alternative to treatment), and
- the methods used to reduce bias and determine sample size.

The overall goal of experimental design is to ensure that the design is appropriate for the objectives of the study. The following design guidelines are intended to indicate what you need to consider when selecting an experimental design.

3.2 Design Guidelines

D1: *Identify the population from which the subjects and objects are drawn.*

If you cannot define the population from which your subject/objects are drawn, it is not possible to draw any inferences from the results of your experiment. This seems to be a particular problem for software engineering surveys. For example, Lethbridge [29] reports a survey where participants were recruited "by directly approaching companies and by advertising on the internet." He states that "The sample appears to be a balanced coverage of a wide spectrum of software professionals, with a bias towards North America—and possibly towards those who were interested enough to take the time to participate." However, the absence of a scientific sampling method means that the results cannot be generalized to any defined population.

Lethbridge's survey can be contrasted with a much more rigorous definition of the population and sampling method given by Ropponen and Lyytinen [40]. They stated "We collected a representative data set using a survey instrument by mailing the developed questionnaire to a selected sample of the members of the Finnish Information Processing Association (1991) whose job title was project manager or equivalent. In order to avoid bias, we sent the questionnaire to at most two persons in one company."

D2: *Define the process by which the subjects and objects were selected.*

The subjects and objects must be representative of the population or you cannot draw any conclusions from the results of your experiment. The most convincing way of obtaining subjects is by random sampling. If a random sample has been obtained, the method of selection should be specified. If the method of selection was not random, it should be explained and you must justify that the sampling method still permits the results of the experiment to generalize to the population of interest.

It is often necessary to define inclusion criteria and exclusion criteria. These are used to identify subjects who will or will not be candidates for participation in the experiment. For example, in software experiments, researchers often use student subjects. In such circumstances, they might choose to exclude any mature students with previous computing experience in order to make sure the participants in the study all have comparable experience.

D3: *Define the process by which subjects and objects are assigned to treatments.*

Subject/objects should be allocated to treatments in an unbiased manner or the experiment will be compromised. It is customary to allocate subjects/objects to treatments by randomization. However, researchers often have small samples in software experiments and simple randomization does not always lead to unbiased allocation nor to equal sample sizes. For example, you might find, by chance, that the two most able subjects are assigned to one treatment and the two least able to another treatment. In such circumstances, a more complex randomization process may be necessary (see, for example, [35], chapter 5), a nonrandomized study may be appropriate [16], or some posttrial adjustment may be necessary to deal with biased groups.

If you decide to use more complex designs or nonrandomized designs, you need to be aware that complex or nonstandard designs may be more difficult to analyze and interpret (see **D4**).

Wilkinson et al. [49] point out that random assignment is sometimes not feasible. In such cases, you need to be particularly careful to identify and control or measure confounding factors and other sources of bias. You must also prepare plans to minimize dropouts and noncompliance with the study protocol and to cope with missing data.

If you intend to perform some post-trial adjustment, you should:

1. Specify how you intend to detect the presence of bias. Such specification usually involves confirming whether the subjects in each of the treatment groups are similar with respect to various characteristics of importance. For example, report whether subjects have similar educational background, or the treatment groups have a similar ratio of male and female subjects.
2. Specify how any bias will be handled during the analysis. For example, to avoid possible bias due to missing values, you may decide to use imputation methods (i.e., methods of replacing missing values with estimated values, see for example [30]).

D4: *Restrict yourself to simple study designs or, at least, to designs that are fully analyzed in the statistical literature. If you are not using a well-documented design and analysis method, you should consult a statistician to see whether yours is the most effective design for what you want to accomplish.*

If you attempt to address the problem of small sample sizes by using more complex designs, you must be sure that you understand all the implications of the selected design. Before developing a new type of experimental design, you should consult an expert. Researchers can easily make mistakes if they attempt to create new designs on their own. In particular,

1. They can produce nested designs where the experimental unit is a group of subjects rather than the individual subject and not realize that this radically changes the nature of the experiment and how it should be analyzed. For example, suppose a researcher in a university wanted to investigate two different methods of testing. Suppose (s)he has four separate tutorial groups each of 10 students. If (s)he assigned each student at random to one of the testing methods, (s)he would have 38 degrees of freedom to test differences between the testing methods. If (s)he assigned each tutorial group as a whole to one of the testing methods, (s)he would have only two degrees of freedom to test any differences between the methods. (In general, the more degrees of freedom, the better the experiment.)
2. They can devise a very complex design that they do not know how to analyze properly. For example, researchers often use cross-over designs to reduce the problem of small sample sizes. In cross-over designs, one experimental unit (team or person) is subjected to several different treatments. However, the cross-over design may have learning or maturation effects (referred to as carryover effects); that is, participation with the first treatment may influence the outcome of the second treatment. For example, a second attempt at debugging or inspection might be easier than the first, regardless of the technique being used. To avoid learning bias, the order in which subjects perform tasks must be randomized or balanced by blocking. Cross-over designs with blocking are more difficult to analyze correctly than researchers may expect (see [33], chapter 32). For instance, when Porter and Johnson [36] reported on two cross-over experiments, the analysis of the first experiment used the Wilcoxon signed rank test. However, this analysis technique cannot be used to analyze a cross-over design of the complexity used in the experiment.

D5: *Define the experimental unit.*

In order to reduce the chance of mistaking the size of experimental unit, identify the experimental unit explicitly when defining your randomization procedure.

Surveys often make mistakes with the experimental unit when questionnaires are sent to multiple respondents in an organization, but the questions concern the organization as a whole. In this case, the unit of observation is the individual, but the experimental unit is the organization. If this problem is not recognized, the total sample size will be incorrectly inflated and the chance of a significant result unfairly increased.

D6: *For formal experiments, perform a pre-experiment or precalculation to identify or estimate the minimum required sample size.*

The sample size determines the probability of finding a difference (by rejecting the null hypothesis) when one exists. For a well-conducted confirmatory experiment, researchers are expected to have information about the expected size of the difference between treatment effects and the variance of the experimental units. From such information, you can calculate the number of subjects (or more correctly, experimental units) required to have a given probability of rejecting the null hypothesis when it is false. This probability is called the power of the test and it is customary to design for a power of at least 0.8. (For more information on sample sizes and appropriate tests, see [5].)

D7: *Use appropriate levels of blinding.*

Medical researchers use double-blind experiments to prevent the participants' expectations (both researchers and subjects) from influencing the study's results. However, it is impossible for participants in a software engineering study not to be aware of the technology they are using. So, it is clear that any expectations about that technology, positive or negative, will influence the study's outcome. For example, the extensive media coverage of the claimed superiority of object-oriented methods suggests that a comparison of object-oriented and structured methods is likely to be biased.

However, several types of blinding are possible in software engineering experiments:

1. Blind allocation of materials. This simply means that the procedure for assigning subjects to treatment groups is separate from the process by which subjects are given any of the materials they will use during the experiment. In software engineering experiments, allocation of subjects and materials can be computerized, thus minimizing the interaction between subjects and researchers during the experiment.
2. Blind marking. Some experimental outcomes are answers to questions that need to be assessed for correctness. If the format of the experimental outcomes is not influenced by the treatment, it is sensible for the scripts to be coded prior to being marked, so the markers cannot tell to which subject or to which treatment group a particular outcome script belongs. For example, if the experiment is concerned with comparing testing methods, and subjects are asked to identify defects in a program, the result of the experiment would be a list of identified defects. The format of the answer would not indicate which testing method had been used, so blind marking would be appropriate.
3. Blind analysis (see **A2**). For blind analysis, the treatments are coded, and the analyst does not know which treatment group is which. Some statisticians believe blind analysis is an effective counter to the problem of fishing for results.

D8: *If you cannot avoid evaluating your own work, then make explicit any vested interests (including your sources of support) and report what you have done to minimize bias.*

Since it is impossible to use double-blinding in software engineering experiments, it is extremely difficult to evaluate any research in which you have a vested interest. For example, if the researchers have themselves created or implemented one of the technologies under evaluation, or if they are champions of a competing technology, there is a strong possibility that their enthusiasm for their own work will bias the trial. Thus, although researchers have a responsibility to identify the hypotheses implicit in their research and provide some preliminary validation of their results, they are not the best people to perform objective, rigorous evaluations of their own technologies. The only direct (but partial) solution to this problem is to employ independent, impartial assessors, taking care to monitor their biases, too. Rosenthal [41] addresses these problems at length and describes several approaches to mitigate experimenter effects.

D9: *Avoid the use of controls unless you are sure the control situation can be unambiguously defined.*

For software experiments, controls are difficult to define because software engineering tasks depend on human skills. Usually, there is no common default technology (i.e., method or procedure or tool) for performing a task against which a new method, procedure, or tool can be compared. That is, we cannot ask one group to use a new design technique and the other to use no design technique. At best, our empirical studies within a particular company can use its standard or current practice as the basis for evaluating a new technology. However, this situation allows us to compare the new technology only with a baseline (i.e., the current industrial practice), not with a defined control. Thus, studies of a new technology in different companies or organizations are difficult to compare because it is highly unlikely that the baselines are the same. For laboratory studies, we can compare two defined technologies, one against the other, but it is usually not valid to compare using a technology with not using it.

D10: *Fully define all treatments (interventions).*

Treatments must be properly defined if experiments are to be replicated and/or their results are to be taken up by industry.

In addition, if we define treatments properly, we can study the same treatment or technique in different organizations. For example, several organizations could collaborate to agree on a common protocol. Such an approach might make it possible to get a better handle on the organization effects as well as the comparison of variables of interest.

D11: *Justify the choice of outcome measures in terms of their relevance to the objectives of the empirical study.*

You must justify that outcome measures are relevant to the objectives of the study to confirm that the design is appropriate to meet the study objectives. The choice of outcome measures is particularly important when they are surrogate measures. For example, researchers wanting to investigate the amount of maintenance effort directed to individual components sometimes count the number of lines of code changed instead of measuring the time software maintainers spend on their maintenance tasks. However, the number of line of code changed is a poor surrogate for maintenance effort and its use in place of task effort should be justified. Inability to measure task effort is not a justification; it is an excuse.

4 Conducting the Experiment and Data Collection

4.1 Introduction

The process of conducting an experiment involves collecting the experimental outcome measures. This is a particular problem for software experiments because our measures are not standardized. Thus, one goal of the data collection guidelines is to ensure that we have defined our data collection process well enough for our experiment to be replicated.

In addition, we need to monitor and record any deviations from our experimental plans. This includes monitoring all dropouts in experiments and nonresponse in surveys.

4.2 Data Collection Guidelines

DC1: *Define all software measures fully, including the entity, attribute, unit and counting rules.*

For empirical software studies, data collection is problematic, in part because, as noted above, software measures are not well-defined. Kitchenham et al. [22] discuss many of the problems with data collection; they suggest several standards for defining and using software measures. From the perspectives of design and data collection, the nonstandard nature of software measures makes it difficult for researchers to replicate studies or to perform meta-analysis of studies of the same phenomenon. That is, without clear standards, we can never be sure that we are measuring the same things or measuring them in the same way. This lack of standards does not mean that everyone should always define and use the same measures. Rather, it means that we need to define measures carefully enough so that we can understand the differences in measurement and, thus, can determine whether we can translate from one measurement scheme to another. For example, it is fine for some researchers or practitioners to measure effort in hours, while others measure it in days; we know how to convert from the one to the other.

DC2: *For subjective measures, present a measure of interrater agreement, such as the kappa statistic or the intraclass correlation coefficient for continuous measures.*

If measures are subjective, the skill and bias of the person determining the measure can affect the results. Empirical studies should discuss the methods used to ensure that measurement is correct and consistent. For data collected by questionnaires, it is necessary to report measures of validity and reliability and other qualities affecting conclusions [49]. For example, researchers have performed studies to investigate the interrater agreement among SPICE model assessors [13].

DC3*: Describe any quality control method used to ensure completeness and accuracy of data collection.*

In order to provide evidence that data collection has been undertaken in an appropriate manner, it is useful to define and report quality control procedures to support the data collection process.

DC4: *For surveys, monitor and report the response rate and discuss the representativeness of the responses and the impact of nonresponse.*

For surveys, it is important to determine the response rate, but it is even more critical to ensure that the nonresponders have not biased the results. It is sometimes possible to sample the nonresponders in order to determine reasons for nonresponse. For example, Ropponen and Lyytinen [40] phoned a random sample of nonrespondents to investigate the reasons why they had not responded. Otherwise, demographic information can be used to justify the representativeness of the responses.

DC5: *For observational studies and experiments, record data about subjects who drop out from the studies.*

Often, subjects who begin a study drop out before the study is complete. For example, a subject may be reassigned to another project, may leave the company, or may refuse to continue to participate. Such situations must be reported carefully in any paper describing the study. It is important to be sure that the dropouts have not biased your results. For example, if all those who dropped out were the most familiar with the baseline technology, then you cannot truly understand the effects of any new technology being evaluated. One means of determining whether or not dropouts have caused systematic bias is to compare dropouts with other subjects on the basis of characteristics such as experience, age, and any preliminary measures available.

Another common example found when undertaking observational studies in industry is that some projects are abandoned before completion. It is important to consider whether incomplete projects will bias any subsequent analysis. This is a particular problem for cost estimation studies, where data about completed projects is used to construct cost models. If a cost model is conditional upon project completion, it has some problems when used to predict costs for a new project. An estimate from a statistically-based cost model is usually assumed to have an implied probability of being achieved (often 0.5 if the model predicts the most likely value). However, if the model has ignored the chance of a project being abandoned before completion, the implied probability will be an overestimate.

DC6: *For observational studies and experiments, record data about other performance measures that may be affected by the treatment, even if they are not the main focus of the study.*

In medical pharmaceutical studies, it is important to record all adverse effects of drugs under test. In software engineering, many of our outcome measures (defect rates, productivity, lead time) are related to each other. Thus, if our main interest is in whether a development method decreases lead time, it may still be important to investigate whether it has adverse or beneficial effects on productivity or defect rates.

5 Analysis

5.1 Introduction

There are two main approaches to analyzing experimental results:

1. Classical analysis (often referred to as the "frequentist" approach). This approach is adopted by most statistical packages.
2. Bayesian analysis. This approach provides a systematic means of making use of "prior information." Prior information may be obtained from previous studies of the phenomenon of interest or from expert opinion.

Our guidelines are independent of the choice of statistical approach. However, Bayesian methods are not usually used in software engineering studies; so, if you decide to adopt a Bayesian approach, you should consult a statistician.

Another fairly general issue is the choice between parametric and nonparametric analysis. If the distribution of variables can be identified, appropriate parametric tests will be more powerful than nonparametric tests. However, if the distribution of variables is unknown, nonparametric methods are usually more appropriate; most are very efficient relative to their parametric counterparts and they are effective with small sample sizes.

Analysis guidelines aim to ensure that the experimental results are analysed correctly. Basically, the data should be analysed in accordance with the study design. Thus, the advantage of doing a careful, well-defined study design is that the subsequent analysis is usually straightforward and clear. That is, the design usually suggests the type and extent of analysis needed. However, in many software engineering examples, the selected design is complex and the analysis method is inappropriate to cope with it. For example, Porter et al. [37] used a cross-over design for subjects but used within subject variance to test the difference between treatments. This is invalid because the resulting *F* test was based on a variance that was smaller and had more degrees of freedom than it should have had. Thus, the researchers were likely to find "statistically significant" results that did not really exist.

5.2 Analysis Guidelines

A1: *Specify any procedures used to control for multiple testing.*

Most software data sets are relatively small and new data are difficult to obtain. Therefore, we have a tendency to overuse our data sets by performing a large number of statistical tests. Multiplicity can be a problem regardless of the size of the data set. Courtney and Gustafson [6] illustrate how performing many statistical tests or making many comparisons on the same dataset can produce a proportionally large number of statistically significant results by chance. (For more information, see [32].) Although their paper comments on correlation studies, the problem is the same for any type of statistical test.

One method for dealing with multiple tests on the same dataset is to adjust the significance level for individual tests to achieve a required overall level of significance as described by Rosenberger [39] or Keppel [21]. For example, if you perform 10 independent tests and require an overall significance level of 0.05, the Bonferroni adjustment requires a significance level of 0.005 for each individual test. Rosenberger describes other, less severe approaches, but each still requires much higher levels of significance for individual tests than the customary 0.05 in order to achieve an overall significance level of 0.05.

The situation regarding multiple tests is made more complicated by the degree to which the comparisons are planned. For example, in many experiments, the researchers have a number of planned comparisons (not too many!), which most statisticians would be willing to test at an individual .05 level (that is, with no Bonferroni adjustment). But, software engineering data are often "messy," in the sense that the relationships among variables are not always clear or strong. If the comparisons are unplanned—the result of a "fishing expedition" to find something to report—then some statisticians insist on using a technique like Bonferroni. Unfortunately, there is no unified view in the statistics community on when a Bonferroni adjustment is necessary.

An alternative to adjusting significance levels is to report the number of results that are likely to have occurred by chance given the number of tests performed. What should be avoided is reporting only positive results with no indication of the number of tests performed. For example, Ropponen and Lyytinen [40] reported 38 significant correlations but did not report how many correlations they tested.

A2: *Consider using blind analysis.*

The problems related to "fishing for results" can be reduced by blind analysis. In a blind analysis, the analysts are given the treatment information in coded format. Since the analysts do not know which treatment is which, they are less likely to over-analyze the dataset in search of a desired result.

A3: *Perform sensitivity analyses.*

Analysts often plunge directly into using a statistical technique without first considering the nature of the data themselves. It is important to look first at the organization of the data, to determine whether any results might be due to outliers or data points that have an unreasonable influence. For example, if all data points save one are clustered around a low value and the one outlier is a very high value, then measures of central tendency may be misleading. It is important to perform a sensitivity analysis to understand how individual data points or clusters of data relate to the behavior of the whole collection.

Lang and Secic [28] identify the following types of sensitivity analysis to consider:

1. Identify and treat outliers.
2. Identify and treat influential observations (which are not necessarily the same as outliers).
3. For regression, assess the independent variables for collinearity.

A4: *Ensure that the data do not violate the assumptions of the tests used on them.*

It is usually necessary to justify the validity of certain assumptions in preparation for analysis. If a particular statistical test relies heavily on a particular distribution, it may be necessary to confirm that the data conform to the required distribution. For instance, the commonly-used Student t test may be inappropriate if the data are heavily skewed, or if the within-group variances are grossly unequal. In cases such as these, it may be appropriate to transform the data (using logarithms, for example) to make them more nearly normal and then check whether the transformation has worked; most statistical software packages can easily apply these transformations and tests.

A5: *Apply appropriate quality control procedures to verify your results.*

No matter what analysis method is used, your first step should be to look at your data. Wilkinson et al. [49] say "Data screening is not data snooping. It is not an opportunity to discard data or change values to favor your hypothesis. However, if you assess hypotheses without examining your data, you risk publishing nonsense." Data screening allows you to check that there are no obviously incorrect values or inappropriate data points in your data set. For example, DePanfilis et al. [8] report analyzing a project data set including function points as a measure of size and effort of completed projects. Investigation of productivity suggested that one project was particularly productive. However, a more detailed review of the project revealed that it had been abandoned before implementation and was, therefore, unsuitable for inclusion in any investigation of productivity. In general, we recommend the use of graphical examination of data and exploratory data analysis [18] before undertaking more detailed analysis.

In addition, most analysis is performed by statistical software. However, you should not assume that statistical software always gives the correct results. If you cannot verify the results using "guesstimates," you should check them against another program [49].

6 Presentation of Results

6.1 Introduction

Presentation of results is as important as the analysis itself. The reader of a study must be able to understand the reason for the study, the study design, the analysis, the results, and the significance of the results. Not only do readers want to learn what happened in a study, but they also want to be able to reproduce the analysis or even replicate the study in the same or a similar context. Thus, design procedures and data collection procedures need to be reported at a level detail that allows the study to be replicated. Analysis procedures need to be described in enough detail that a knowledgeable reader with access to the original data would be able to verify the reported results and test the stated hypotheses.

We will not repeat all the preceding context, design, analysis, and data collection guidelines restating them as presentation guidelines, rather we restrict ourselves to issues that are directly related to presentation and have not been covered previously.

We should also keep in mind that a particular study may be combined with others in a meta-analysis. Consequently, authors should include information that would support such analysis in the future.

6.2 Presentation Guidelines

P1: *Describe or cite a reference for all statistical procedures used.*

Following the analysis guidelines, it is important to document all statistical procedures. Reference to a statistical package is **not** sufficient. However, there are some exceptions to that rule. For example, Fukuda and Ohashi [12] suggest that the following statistics do not require explicit references: *t* test, simple chi-squared test, Wilcoxon or Mann-Whitney U-test, correlation, and linear regression.

P2: *Report the statistical package used.*

Statistical packages often give slightly different results. So, it is important to specify which statistical package has been used.

P3: *Present quantitative results as well as significance levels. Quantitative results should show the magnitude of effects and the confidence limits.*

McGuigan [31] and Altman [2] both identified many common statistical errors in medical papers, related to failing to report results at an appropriate level of detail. Combining their lists of errors with the Lang and Secic [28] guidelines for reporting inferential statistics, we have compiled the following checklist:

1. Report information about within person difference when using paired data.
2. Report the magnitude of an effect size.
3. Report confidence limits for inferential statistics including means, mean differences, correlation coefficients and regression coefficients.
4. Report the alpha level used for statistical testing
5. Report whether the tests were two-tailed or one-tailed.
6. Report the value of the t statistics.
7. For regression, report the regression equation.
8. For regression, report the coefficient of determination.
9. For regression, if the model is to be used for prediction, report the validation procedure and results.
10. To support the requirements of meta analysis, always report the standard error of the mean change in outcome measures when measures change from the baseline to a later time.

P4: *Present the raw data whenever possible. Otherwise, confirm that they are available for confidential review by the reviewers and independent auditors.*

It is essential to present the raw data as well as the analysis and results. Yancey [50] criticizes papers painting only a partial picture, pointing out that "the reader's only consult with nature is via the data reported in the article. Everything else is consult with authority. That is not science." Unfortunately, many empirical software-related studies are based on real project data that cannot be published because of its commercial sensitivity. Although it is unrealistic to reject any paper that does not publish its data, it is important to remember that software engineering must be founded on science, not anecdote. Thus, we recommend that, when the data cannot be published outright, the authors make raw data available to reviewers on a confidential basis, or that raw data be made available to independent auditors on the same confidential basis.

P5: *Provide appropriate descriptive statistics.*

Lang and Secic [28], McGuigan [31], and Altman [2] identified a number of problems with simple descriptive statistics. In particular, they were concerned that the measures of central tendency and dispersion were often inappropriate. For example, means and standard deviations were reported for ordinal variables and heavily skewed interval/ratio variables. They were also concerned about spurious precision. Combining their advice, we have compiled the following checklist defining which descriptive statistics should be reported and how they should be reported:

1. Report the number of observations.
2. Report all numbers with the appropriate degree of precision, e.g., means no more than one decimal place more than the raw data.
3. Present numerator and denominator for percentages.
4. With small numbers, present values not percentages.
5. Present appropriate measures of central tendency and dispersion when summarizing continuous data.
6. Do not use the standard error in place of the standard deviation as a measure of dispersion.

7. If continuous data have been separated by "cutpoints" into ordinal categories, give the rationale for choosing them.
8. If data have been transformed, convert the units of measurement back to the original units for reporting.

P6: *Make appropriate use of graphics.*

Graphical representations of results are often easier for readers to understand than complicated tables. However, graphical presentations have to be undertaken carefully, they can also be misleading. Common graphical errors are:

1. Representing one-dimensional data in two or more dimensions.
2. Using Pie charts (which are inherently less accurate than alternative displays).
3. Inappropriate choice of scale to emphasize things that support the conclusions or de-emphasize things that do not support the conclusions.
4. Omitting outlying points from scatter plots.
5. Omitting jittering on scatter plots when many data points overlap.

7 Interpretation of Results

7.1 Introduction

The main aim for the interpretation or conclusions section of a paper is that any conclusions should follow directly from the results. Thus, researchers should not introduce new material in the conclusions section.

It is important that researchers do not misrepresent their conclusions. For example, it is easy to play down the significance of findings that conflict with previous research. It is also important that researchers qualify their results appropriately.

7.2 Interpretation guidelines

I1: *Define the population to which inferential statistics and predictive models apply.*

This follows directly from design guideline **D1**. If the population is not well-defined, we cannot interpret any inferential statistics used in the analysis, nor can we be sure how any predictive models could be used in practice. Thus, the results of the experiment are unlikely to be of practical value.

I2: *Differentiate between statistical significance and practical importance.*

The study design suggests the types of analysis appropriate for the situation, the problem, and the data. However, the researchers must interpret the results in the context of these elements plus the findings of others doing related work. It is important to differentiate statistical significance from practical importance. That is, research may show a statistical significance in some result, but there may be no practical importance. Confidence intervals can help us in making this determination, particularly when statistical significance is small. That is, first see whether the result is real (statistically significant); then see whether it matters (practical significance). For example, with a large enough dataset, it is possible to confirm that a correlation as low as 0.1 is significantly different from 0. However, such a low correlation is unlikely to be of any practical importance.

In some cases, even if the results are not statistically significant, they may have some practical importance. Confidence limits can support assertions regarding the importance of nonsignificant results. Power analyses can also be used to suggest a reason for lack of significance. However, you should not use the excuse of low power to increase the alpha level post hoc. Hypothesis-testing involves three interdependent variables, power, significance level, and sample size, which determine the test. Either significance level and sample size are fixed and the power is to be determined, or significance level and power are fixed, and the sample size is calculated.

I3: *Define the type of study.*

Several guidelines identify the need to specify the type of study [4], [12], [49]. This is for two reasons:

1. To establish the reliance that readers should put on the conclusions of the study. Wilkinson et al. say "Do not cloak a study in one guise to give it the assumed reputation of another" [49].
2. To suggest the suitability of the study for meta-analysis.

Zelkowitz and Wallace [51] listed 12 experimental methods used in software engineering studies identifying their relative strengths and weaknesses. The 12 methods were classified into three categories: observational, historical, and controlled. Fukuda and Ohashi [12] present a more detailed taxonomy of experimental types as follows:

1. Observational studies which can be of three main types: Case series study, Cross-sectional study, or Longitudinal study. Longitudinal studies are of three types: a) Case-control study, b) Cohort study (prospective, retrospective, historical), and c) Nested case-control study.
2. Intervention study (trial) which can be of two main types: Controlled or Uncontrolled. Controlled studies are three main types: a) Parallel (randomized, nonrandomized), b) Sequential (self-controlled, cross-over), and c) Historical control.

It is important to distinguish between confirmatory studies, from which strong conclusions can be drawn, and exploratory studies, from which only weak conclusions can be drawn. For example, Yancey [50] states that "Only truly randomized tightly controlled prospective studies provide even an opportunity for cause-and-effect statements." He suggests that authors of less rigorous studies need to point out that only weak conclusions can be drawn from their results.

It is particularly dangerous to rely heavily on regression and correlation studies because they are usually exploratory in nature and do not identify causal relationships. Without causal relationships the value of regression models is reduced. For example, Hitz and Montazari [17] point out that "While a *prediction* model can be based on the more general statistical relationship, a model used *to control* the development process should be based on causal relationships."

I4: *Specify any limitations of the study.*

Researchers have a responsibility to discuss any limitations of their study. They usually need to discuss at least internal and external validity. Internal validity relates to the extent to which the design and analysis may have been compromised by the existence of confounding variables and other unexpected sources of bias. External validity relates to the extent to which the hypotheses capture the objectives of the research and the extent to which any conclusions can be generalized.

It is encouraging that recent software research papers have included discussion of threats to validity (see, for example, [19], [26]). However, studies still report that some projects were omitted from the analysis because "they did not collect all the required data," but do not consider the implication of those projects on their conclusions. If the authors were looking for quality differences among projects, it would be reasonable to assume that there are more quality problems with projects that were unable to collect data than those that were. So, they may have difficulty finding the evidence for which they were searching.

Other general validity problems that affect formal experiments are the use of students as subjects and the choice of software engineering materials. Some practitioners may feel the use of student subjects in formal experiments reduces the practical value of experiments. In our view, this is not a major issue as long as you are interested in the evaluating the use of a technique by novice or nonexpert software engineers. Students are the next generation of software professionals and, so, are relatively close to the population of interest. This can be contrasted with the use of students in psychology studies as representatives of the human population as a whole. The problem of the choice of materials is more problematic. Formal experiments inevitably try out techniques on restricted problems with known solutions, it is impossible to be sure that techniques evaluated under such circumstances will scale up to industrial size systems or very novel programming problems.

8 Conclusions and Discussion

We have presented several guidelines that we hope will improve the quality of performing and evaluating empirical research in software engineering. The guidelines are based on our own experience as researchers and reviewers, and on recommendations from other disciplines whose advancement requires careful empirical study. We believe that guidelines are important because:

1. In our experience, supported by the examples in this paper, software researchers often make statistical mistakes.
2. At they same time, senior researchers are pressing for more empirical research to underpin software engineering [51], [45].

We do not pretend to be immune to the practices we criticize. In fact, our search for guidelines outside of software engineering was prompted not only by our discomfort with studies reported in the papers we read but also by problems with our own research. For example, we often fail to define explicitly the population to which our inferential and statistics and predictive models apply. And many of us are guilty of failing to adjust statistical significance levels when performing many statistical tests (see, for example, [42]).

We present these guidelines for use by all of us. Researchers can improve their research planning, implementation, and reporting. Reviewers and readers can use the guidelines to judge the quality of the research. And the guidelines are essential for those planning to assess studies that are to be combined in a meta-analysis. Some of our guidelines have ethical as well as methodological implications. We have not emphasized this issue. However, serious ethical issues arise in evaluating your own work, ignoring the dangers both of dropping observations and of multiple testing, and misrepresenting findings, for example, failing to report data that are not in line with expectation. Rosenthal [42] presents a more detailed discussion of the relation between scientific quality and ethics in research.

Our guidelines represent a starting point for discussion. We do not suggest that these guidelines are complete, nor that they will solve all the problems associated with empirical software engineering research. In particular, these guidelines alone will not improve the relevance and usefulness of empirical software engineering research. They must be combined with careful consideration of the implication of the outcomes of research. That is we do not want to do research for the sake of research. We share this problem with medicine and many other disciplines. Altman [2] says "Sadly, much research may benefit researchers rather more than patients, especially when it is carried out primarily as a ridiculous career necessity."

In addition, we do not believe the guidelines to be sufficient by themselves. It is important for editorial boards of software engineering journals and conferences to take a lead in this issue. For example, we believe that editorial boards should:

1. Publish guidelines or checklists for reviewing papers containing study designs, implementation descriptions, and statistical analysis.
2. Ensure that empirical studies are reviewed by experienced statisticians.
3. Commission periodic systematic reviews of the quality of statistics in papers appearing in their journals.

Furthermore, we believe that journals and conferences should adopt a clear policy concerning the need for presenting raw data; they should also identify the conditions under which papers will be published without raw data.

Acknowledgments

This paper arose from the project "Methods for Valid Analysis of Software Datasets" funded by the United Kingdom Engineering and Physical Sciences Research Council (GR/L 28371). The authors also thank Richard Kemmerer, former editor-in-chief of *IEEE Transactions on Software Engineering*, for his encouragement. The appendix to this paper can be viewed for free on our digital library at http://computer.org/tse/archives.htm.

References

[1] D. Altman, "Guidelines for Contributors," *Statistics in Practice,* S.M. Gore and D. Altman, eds., 1991.

[2] D. Altman, "Statistical Reviewing for Medical Journals," *Statistics in Medicine,* vol. 17, pp. 2661-2674, 1998.

[3] D. Altman, S. Gore, M. Gardner, and S. Pocock, "Statistical Guidelines for Contributors to Medical Journals," *British Medical J.,* vol. 286, pp. 1489-1493, 1983.

[4] C. Begg, M. Cho, S. Eastwood, R. Horton, D. Moher, I. Olkin, R. Pitkin, D. Rennie, K.F. Schultz, D. Simel, and D.F. Stroup, "Improving the Quality of Reporting of Randomized Trials (the CONSORT Statement)," *J. Am. Medical Association,* vol. 276, no. 8, pp. 637-639, Aug. 1996.

[5] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences,* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1988.

[6] R.E. Courtney and D.A. Gustafson, "Shotgun Correlations in Software Measures," *Software Eng. J.,* vol. 8, no. 1, pp. 5-13, 1992.

[7] L.J. Cronbach, "Coefficient Alpha and the Internal Structure of Tests," *Pscychometrika,* vol. 16, no. 3, pp. 297-334, 1951.

[8] S. DePanfilis, B. Kitchenham, and N. Morfuni, "Experiences Introducing a Measurement Program," *Information and Software Technology,* vol. 39, no. 11, pp 745-754, 1997.

[9] E. Doolan, "Experience with Fagan's Inspection Method," *Software—Practice and Experience,* vol. 22, no. 2, pp. 173-182, Feb. 1992.

[10] K. El-Emam, S. Benlarbi, N. Goel, and S. Rai, "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics," *IEEE Trans. Software Eng.,* vol. 27, no. 6, pp. 630-650, July 2001.

[11] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach,* second ed. Brooks-Cole, 1997.

[12] H. Fukuda and Y. Ohashi, "A Guideline for Reporting Results of Statistical Analysis in Japanese Journal of Clinical Oncology," *Japanese J. Clinical Oncology,* vol. 27, pp. 121-127, 1997, also available in English at http://wwwinfo.ncc.go.jp/jjco/.

[13] P. Fusaro, K. El-Emam, and B. Smith, "Evaluating the Interrater Agreement of Process Capability Ratings," *Proc. Fourth Int'l Software Metrics Symp.,* pp. 2-11, 1997.

[14] M.J. Gardner and D.G. Altman, *Statistics with Confidence,* London: BMJ, 1989.

[15] L. Gordis, *Epidemiology.* W.B. Sunders Company, 1996.

[16] D. Heinsman and W. Shadish, "Assignment Methods in Experimentation: When Do Nonrandomized Experiments Approximate Answers from Randomized Experiments?" *Psychological Methods,* vol. 1, no. 2, pp. 154-169, 1996.

[17] M. Hitz and B. Montazeri, "Measuring Product Attributes of Object-Oriented Systems," *Proc. Fifth European Software Eng. Conf.,* W. Schafer and P. Botella, eds., Sept. 1995.

[18] D.C. Hoaglin, F. Mosteller, and J.W. Tukey, *Understanding Robust and Exploratory Data Analysis.* John Wiley, 1983.

[19] M. Host and C. Wohlin, "A subjective Effort Estimation Experiment," *Information and Software Technology,* vol. 39, pp. 755-762, 1997.

[20] P. Johnson and D. Tjahjono, "Does Every Inspection Really Need a Meeting?" *Empirical Software Eng.,* vol. 3, pp. 9-35, 1998.

[21] G. Keppel, *Design and Analysis: A Researcher's Handbook,* third ed. Prentice Hall, 1991.

[22] B.A. Kitchenham, R.T. Hughes, and S.G. Linkman, "Modeling Software Measurement Data," *IEEE Trans. Software Eng.,* vol. 27, no. 9, pp. 788-804, Sept. 2001.

[23] B.A. Kitchenham, G.H. Travassos, A. Von Mayrhauser, F. Niessink, N.F. Schniedewind, J. Singer, S. Takado, R. Vehvilainen, and H. Yang, "Towards an Ontology of Software Maintenance," *J. Software Maintenance: Research and Practice,* vol. 11, pp. 365-389, 1999.

[24] B.A. Kitchenham, S.L. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation," *IEEE Trans. Software Eng.,* vol. 21, no. 12, pp. 929-944, Dec. 1995.

[25] B.A. Kitchenham and K. Kansala, "Inter-Item Correlations Among Function Points," *Proc. First Int'l Software Metrics Symp.,* pp. 11-14, 1993.

[26] O. Laitenberger and J.-M. DeBaud, "Perspective-Based Reading of Code Documents at Robert Bosch GmbH," *Information and Software Technology,* vol. 39, pp. 781-791, 1997.

[27] L. Land, C. Sauer, and R. Jeffery, "Validating the Defect Detection Performance Advantage of Group Designs for Software Reviews: Report of a Laboratory Experiment Using Program Code," *Proc. Sixth European Software Eng. Conf.,* M. Jazayeri and H. Schauer, eds., pp. 294-309, 1997.

[28] T. Lang and M. Secic, *How to Report Statistics in Medicine: Annotated Guidelines for Authors, Editors and Reviewers.* Am. College of Physicians, 1997.

[29] T.C. Lethbridge, "What Knowledge is Important to a Software Professional?" *Computer,* vol. 33, no. 5, pp. 44-50, May 2000.

[30] R. Little and D. Rubin, *Statistical Analysis With Missing Data.* John Wiley & Sons, 1987.

[31] S.M. McGuigan, "The Use of Statistics in the *British Journal of Psychiatry*," *British J. Psychiatry,* vol. 167, no. 5, pp. 683-688, 1995.

[32] R.G. Miller, Jr., *Simultaneous Statistical Inference,* second ed. New York: Springer-Verlag, 1981.

[33] G.A. Milliken and D.A. Johnson, *Analysis of Messy Data, Volume 1: Designed Experiments.* London: Chapman & Hall, 1992.

[34] L.M. Pickard, B.A. Kitchenham, and P. Jones, "Combining Empirical Results in Software Engineering," *Information and Software Technology,* vol. 40, no. 14, pp. 811-821, 1998.

[35] S.J. Pocock, *Clinical Trials: A Practical Approach.* Chichester, U.K.: John Wiley and Sons, 1984.

[36] A.A. Porter and P.M. Johnson, "Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies," *IEEE Trans. Software Eng.,* vol. 23, no. 3, pp. 129-145, Mar. 1997.

[37] A.A. Porter, L.G. Votta, and V.R. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," *IEEE Trans. Software Eng.,* vol. 21, no. 6, pp. 563-575, June 1995.

[38] A.M. Porter, "Misuse of Correlation and Regression in Three Medical Journals," *J. Royal Soc. Medicine,* vol. 92, no. 3, pp. 123-128, 1999.

[39] W.F. Rosenberger, "Dealing with Multiplicities in Pharmacoepidemiologic Studies," *Pharmacoepidemiology and Drug Safety,* vol. 5, pp. 95-100, 1996.

[40] J. Ropponen and K. Lyytinen, "Components of Software Development Risk: How to Address Them? A Project Manager Survey," *IEEE Trans. Software Eng.,* vol. 26, no. 2, pp. 98-111, 2000.

[41] R. Rosenthal, *Experimenter Effects in Behavioral Research,* New York: John Wiley and Sons, 1976.

[42] R. Rosenthal, "Science and Ethics in Conducting, Analyzing, and Reporting Psychological Research," *Psychological Science,* vol. 5, pp. 127-134, 1994.

[43] H.S. Sacks, J. Berrier, D. Reitman, V.A. Ancona-Berk, and T.C. Chalmers, "Meta-Analyses of Randomized Controlled Trials," *The New England J. Medicine,* vol. 316, no. 8, pp. 312-455, Feb. 1987.

[44] H. Siy and L. Votta, "Does the Modern Code Inspection Have Value?" *Proc. IEEE Int'l Conf. Software Maintenance,* 1999.

[45] W.F. Tichy, "Should Computer Scientists Experiment More?," *Computer,* vol. 31, no. 5, pp. 32-40, May 1998.

[46] R. Vinter, M. Loomes, and D. Kornbrot, "Applying Software Metrics to Formal Specifications: A Cognitive Approach," *Proc. Fifth Int'l Software Metrics Symp.,* pp. 216-223, 1998.

[47] L. Votta, "Does Every Inspection Need a Meeting?" *ACM Software Eng. Notes,* vol. 18, no. 5, pp. 107-114, 1993.

[48] G.E. Welch and S.G. Gabbe, "Review of Statistics Usage in the *American Journal of Obstetrics and Gynecology*," *Am. J. Obstetrics and Gynecology,* vol. 175, no. 5, pp. 1138-1141, 1996.

[49] L. Wilkinson and Task Force on Statistical Inference, "Statistical Methods in Psychology Journals: Guidelines and Explanations," *Am. Psychologist,* vol. 54, no. 8, pp. 594-604, 1999, (http://www.apa.org/journals/amp/amp548594.html).

[50] J.M. Yancey, "Ten Rules for Reading Clinical Research Reports," *Am. J. Orthodontics and Dentofacial Orthopedics,* vol. 109, no. 5, pp. 558-564, May 1996.

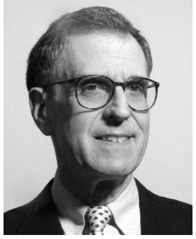
[51] M.V. Zelkowitz and D.R. Wallace, "Experimental Models for Validating Technology," *Computer,* vol. 31, no. 5, pp. 23-31, May 1998.

Barbara A. Kitchenham received the PhD degree from the University of Leeds. She is a professor of quantitative software engineering at Keele University. Her main research interest is software metrics and its application to project management, quality control, risk management, and evaluation of software technologies. She is particularly interested in the limitations of technology and the practical problems associated with applying measurement technologies and experimental methods to software engineering. She is a chartered mathematician and fellow of the Institute of Mathematics and Its Applications. She is also a fellow of the Royal Statistical Society. She is a visiting professor at both the University of Bournemouth and the University of Ulster. She is a member of the IEEE Computer Society.

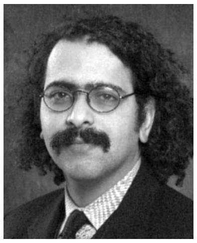
Shari Lawrence Pfleeger is the president of Systems/Software, Inc., a consultancy specializing in software engineering and technology. From 1997 to 2000, she was a visiting professor at the University of Maryland's computer science department. Also, she was the founder and director of Howard University's Center for Research in Evaluating Software Technology (CREST), and was a visiting scientist at the City University (London) Centre for Software Reliability, principal scientist at MITRE Corporation's Software Engineering Center, and manager of the measurement program at the Contel Technology Center. She began her career as a developer and maintainer for real-time, business-critical software systems. Thus, she has experience both with the practical problems of software development and the theoretical underpinnings of software engineering and computer science. For several years, Dr. Pfleeger was the associate editor-in-chief of *IEEE Software*, where she edited the Quality Time column. She is currently an associate editor of *IEEE Transactions on Software Engineering*, and she is a member of the editorial board of Prentice Hall's Software Quality Institute series. She is a member of IEEE, the IEEE Computer Society, and the Association for Computing Machinery. Dr. Pfleeger was on the executive committee of the Technical Council on Software Engineering from 1996 to 2000.

Lesley M. Pickard received the PhD degree from City University, London, on the use of software metrics to support project management and is a fellow of the Royal Statistical Society. She is a research fellow at Keele University. Her main research interests are in software measurement and their use in supporting project and risk management. She is interested in the identification of valid statistical techniques and their limitations when used in software engineering.

Peter W. Jones is currently a professor of statistics in the Mathematics Department at Keele University, UK. Previously, he was a senior lecturer and reader in mathematical statistics at Keele and a lecturer in statistics at the University of Wales, Aberystwyth where he gained his PhD in 1974. He is also a nonexecutive director on the board of the Orthopaedic Hospital in Oswestry, UK. Peter is an associate editor of the *Journal Biometrics*, a member of the international advisory board of the *Egyptian Journal of Mathematics and Statistics*, and an editor of the *Journal of Apicultural Research*. His research interests cover a wide range from theoretical developments in Bayesian sequential estimation and optimisation, especially the study of bandit processes, to applications in medicine. In recent years, his work has concentrated on modeling genetic effects on susceptibility and severity of disease, meta analysis, the planning of clinical trials, and the development of clinical scoring models for disease progression. He is the author or coauthor of more than 200 papers.



David C. Hoaglin received the PhD degree in statistics from Princeton University in 1971. He is a principal scientist at Abt Associates Inc., an applied social research firm in Cambridge, Massachusetts. He has long been interested in statistical computing and in diverse aspects of software. His other research intetests include exploratory data analysis, outliers, nonresponse in surveys, and applications of statistics to policy problems. In recent years, he has been involved in conducting the US National Immunization Survey. Earlier, he held faculty and research positions in the Department of Statistics at Harvard University. He is the coeditor and coauthor (with Frederick Mosteller and John Tukey) of the books *Understanding Robust and Exploratory Data Analysis*; *Exploring Data Tables, Trends, and Shapes*; and *Fundamentals of Exploratory Analysis of Variance*. He is a fellow of the American Statistical Association, a fellow of the American Association for the Advancement of Science, and an ordinary member of the International Statistical Institute.



Khaled El Emam obtained the PhD degree from the Department of Electrical and Electronics Engineering, King's College, the University of London (UK) in 1994. He is currently a research officer at the National Research Council in Ottawa, Canada's primary applied research organization. He is coeditor of ISO's project to develop an international standard defining the software measurement process (ISO/IEC 15939) and is leading the software engineering process area in the IEEE's project to define the Software Engineering Body of Knowledge. He has also coedited two books on software process, both published by the IEEE CS Press; he is an adjunct professor at both the School of Computer Science at McGill University and the Department of Computer Science at the University of Quebec at Montreal; and is president of a software company specializing in online analytics. Currently, he is a resident affiliate at the Software Engineering Institute in Pittsburgh. Khaled is on the editorial boards of *IEEE Transactions on Software Engineering* and the *Empirical Software Engineering Journal*. Previously, he was the international trials coordinator for the SPICE Trials, where he was leading the empirical evaluation of the emerging process assessment International Standard, ISO/IEC 15504, world-wide; the head of the Quantitative Methods Group at the Fraunhofer Institute for Experimental Software Engineering in Germany; a research scientist at the Centre de recherche informatique de Montreal (CRIM) in Canada; a researcher in the software engineering laboratory at McGill University; and worked in a number of research and development projects for organizations, such as Toshiba International Company and Honeywell Control Systems in the UK, and Yokogawa Electric in Japan. He is a member of the IEEE Computer Society.

Jarrett Rosenberg received his doctorate from the University of California at Berkeley and worked at Xerox and Hewlett-Packard Laboratories before joining Sun in 1990. He is a statistician and quality engineer at Sun Microsystems. He is a member of the ACM, the American Statistical Association, and the American Society for Quality, and is an ASQ Certified Quality Engineer, Reliability Engineer, and Software Quality Engineer. He also is a member of the IEEE Computer Society.