An Empirical Investigation of the Influence of a Type of Side Effects on Program Comprehension

J.J. Dolado, *Member*, *IEEE*,
M. Harman, *Member*, *IEEE*,
M.C. Otero, and L. Hu

Abstract—This paper reports the results of a study on the impact of a type of side effect (SE) upon program comprehension. We applied a crossover design on different tests involving fragments of C code that include increment and decrement operators. Each test had an SE version and a side-effect-free (SEF) counterpart. The variables measured in the treatments were the number of correct answers and the time spent in answering. The results show that the side-effect operators considered significantly reduce performance in comprehension-related tasks, providing empirical justification for the belief that side effects are harmful.

Index Terms—Side-effect-free programs, crossover designs, program comprehension.

1 Introduction

A side effect is any change in program state that occurs as a by-product of the evaluation of an expression. Side effects are often thought to impede program comprehension, although hitherto, this appears to be a belief that has not been examined empirically. This paper is concerned with a type of side effect in C programs and their effect upon program comprehension.

The C programming language standard defines side effects in a very broad manner ([9, Section 5.1.2.3]): "Accessing a volatile object, modifying an object, modifying a file, or calling a function that does any of those operations are all side effects, which are changes in the state of the execution environment. Evaluation of an expression may produce side effects." In this paper, we adopt a more restricted view, considering only auto-increment and auto-decrement operators. A side effect will be taken to be: "Any change in the value of a variable which occurs when an expression is evaluated, other than an assignment expression-statement."

A side-effect-free program is one which contains no side effects. This defines side effects to occur in expressions like "`x++`," "`- -v`," "`(x+1,y=x+3)`," and "`x==2 && y=2*x`." Assignment statements in C are assignment expression-statements. To be side-effect free, assignment expressions are allowed to change only the value of the variable on the lefthand side of the assignment; the righthand side must not change the value of any variables.

Our basic research enquiry is centered around the question: "Are side-effect-free programs more understandable than programs with side-effects?" This paper uses a side effect removal algorithm, LinSERT [6], to produce side-effect-free versions of C program fragments which contain side effects. The tool is used to avoid the potential source of bias inherent in human selection of side-effect-free programs. LinSERT's algorithm was designed to remove side effects, not to produce readable code. Indeed, there are cases when, subjectively, one could argue that the results obtained by LinSERT are not ideally readable. However, this observation only serves to strengthen the results obtained.

- J.J. Dolado and M.C. Otero are with the Department of Computer Languages and Systems, University of the Basque Country, 20.009-San Sebastián, Spain. E-mail: jipdocoj@si.ehu.es, jipotvim@vc.ehu.es.
- M. Harman and L. Hu are with the Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK. E-mail: {mark.harman, Lin.Hu}@brunel.ac.uk.

Manuscript received 28 May 2002; revised 12 Mar. 2003; accepted 17 Mar. 2003.
Recommended for acceptance by G. Canfora.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 116634.

0098-5589/03/$17.00 © 2003 IEEE Published by the IEEE Computer Society

1.1 Related Work

Side effects are widely believed to inhibit program understanding, with a consequent detrimental impact upon software maintenance and evolution. For example, Kernighan and Pike [10] suggest that side effects should only be used in very special situations, where well-understood side effect idioms are employed to improve a program's performance. Other authors, for example, [1], advise caution, suggesting that the programmer should carefully consider "the tradeoff between increased speed and decreased maintainability that results when embedded assignments are used in artificial places."

The study of side effects is related to studies concerning the effects on program comprehension of the syntactic presentation of the program code, which in turn affects readability. Miara et al. [15] showed that indentation has a significant effect on comprehension for both experienced and novice programmers. Oman and Cook [19] studied the effect on program comprehension of source code formatting and commenting. Related to the issue of studying a language construct, we may be in a similar situation to other assertions about potential dangers of some practices. The first of the "considered harmful" saga began with the seminal work of Dijkstra [4], which spun a new and fresh approach to programming, as well as a decade of fruitful discussions. The type of side effects that we are dealing with here can be also labeled as one in the list of activities suggested "harmful." In some cases, the practices have the form of a "taboo" [14]. The activity of reading code plays a basic role in the development and maintenance activities; therefore, any improvement in the procedures, methods, languages, and tools for increasing the comprehensibility of the code will have direct effects on the progress of the software building activities [7].

In the rest of the paper, we present the details of the experiment and its analysis. In Section 2, we describe the experiment, the design, hypotheses, and other elements. Section 3 presents the analysis of the data. Finally, Section 4 states the conclusions.

2 Formal Experiment

2.1 Crossover Designs for the Experiment

We considered two types of programs, SEF versus SE programs, and we planned a parallel study between two groups. In order to have the two groups getting both treatments (SEF and SE), we performed a crossover design in which the two groups received both treatments but in different order. Applying just one treatment for each subject would have the drawback that the variation of the measurements between subjects could distort the true effect of the treatment. By taking measures of each individual in both treatments, we avoid this variability, although we may incur other problems, such as detecting other effects apart from the treatment.

Crossover designs, in which each subject receives a sequence of treatments (repeated measures), are a well-known type of experimental design, used in clinical and medical studies [5], [8], [12]. The most widely used design of this type is the two treatment, two-period (two-round) crossover design. This structure represents, for instance, first, the administration of two drugs to the two groups and then, after withdrawal, the reverse application of the treatments. However, these well-known designs present several problems, both in their use of terminology and in their analysis of data [3].

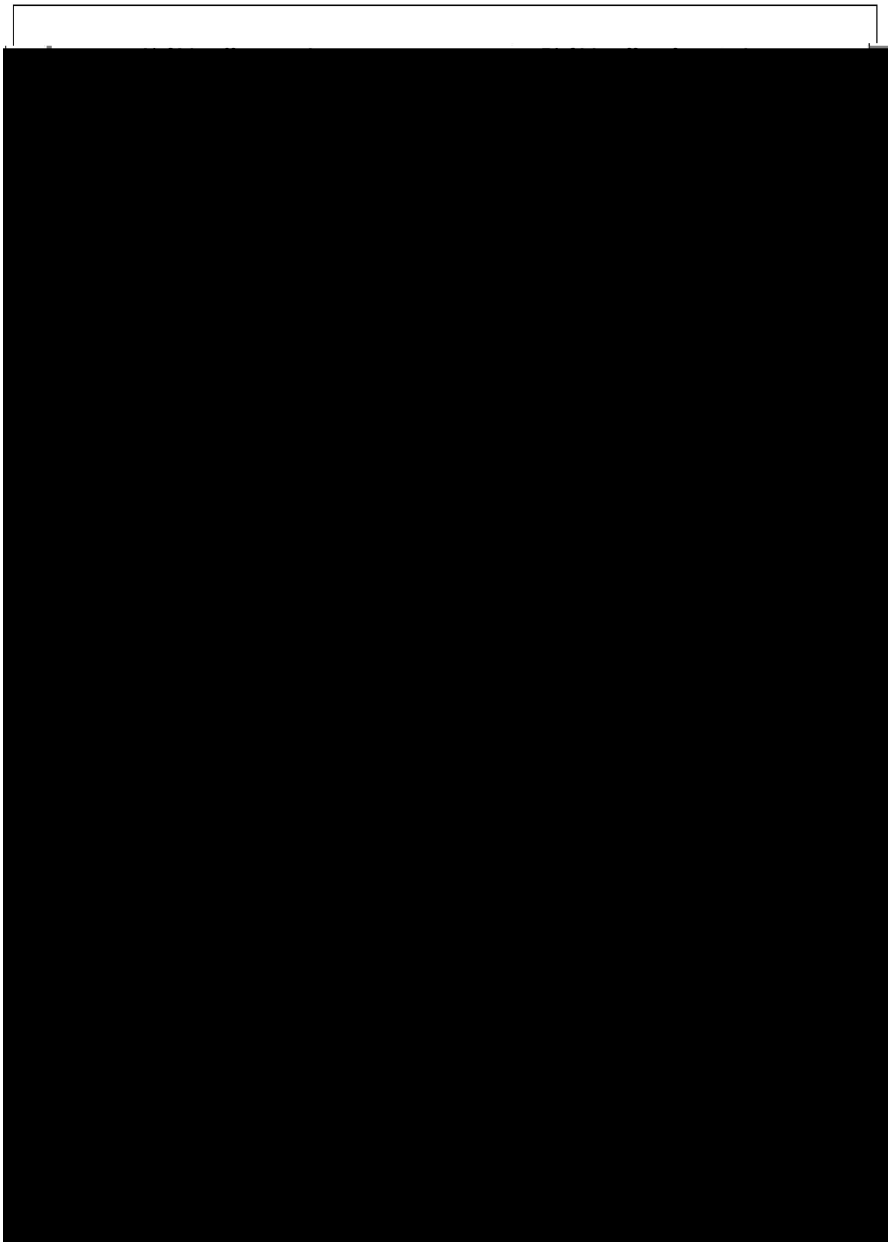

Fig. 2. Sample questions in the tests (both versions).

for Test 2, we test the null hypotheses $H_{0-T2-SCO-TREAT}$ and $H_{0-T2-TIM-TREAT}$.

2. Null hypotheses for the periods and for the groups: We state that there is no significant difference between the means of the SCOREs obtained in the tests with respect to the two different PERIODs of treatment ($H_{0-T1-SCO-PERIOD}$) and ($H_{0-T2-SCO-PERIOD}$) and with respect to the two GROUPs (equivalent to testing the CARRYOVER effects, $H_{0-T1-SCO-GROUP}$, and $H_{0-T2-SCO-GROUP}$). The same null hypotheses are stated for the TIMEs in both tests ($H_{0-T1-TIM-PERIOD}$, $H_{0-T2-TIM-PERIOD}$, $H_{0-T1-TIM-GROUP}$, and $H_{0-T2-TIM-GROUP}$).

Test 3 has the corresponding set of hypotheses.

2.6 Power Analysis and Sample Size

As prescribed in experimental design, the identification of the sample size, the effect size, and the desired power is a requirement before conducting the experiment [11], [13], [17]. We performed an "a priori" power analysis, exploring the relationships among the sample size (n), the effect size (f), the significance level (α), and the desired power ($1 - \beta$) since we could not change, for instance, the sample size n.

We have to remember that α is the probability of committing a type I error. That is, to reject, incorrectly, the null hypothesis when it is true. As the process of transforming SE into SEF is performed automatically, the cost that we would incur when committing a type I error can be safely downsized (no human cost is involved in the transformation process). The type I error means to assume that SEF and SE programs have different effects when they actually have not. When automation is unavailable, the cost involved is the additional effort, if there is any, of side-effect-free coding. On the other hand, committing a type II error, that is, to accept, incorrectly, the null hypothesis when it is false, would imply that we will be wasting human effort as we would be using the SE versions instead of taking advantage of their SEF counterparts. The commission of a type II error, represented by β, means to

Sevilla, and the anonymous referees for their invaluable comments. The experimental material and the data used in this article can be found at http://www.sc.ehu.es/jiwdocoj/sef/sef.htm.

REFERENCES

[1] L. Cannon, R. Elliott, L. Kirchhoff, J. Miller, J. Milner, R. Mitze, R. Schan, E. Whittington, N. Spencer, H. Keppel, D. Brader, and M. Brader, "Recommended C Style and Coding Standards," http://www.cs.umd.edu/users/cml/cstyle/indhill-cstyle.html, 2000.

[2] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, second ed. Lawrence Erlbaum Assoc., 1988.

[3] G.E. Dallal, "The Computer-Aided Analysis of Crossover Studies," http://www.tufts.edu/~gdallal/crossovr.htm, 2000.

[4] E.W. Dijkstra, "Goto Statement Considered Harmful," Comm. ACM, vol. 11, no. 3, pp. 147-148, Mar. 1968.

[5] J.E. Grizzle, "The Two-Period Change-Over Design and Its Use in Clinical Trials," Biometrics, vol. 21, pp. 467-480, 1965.

[6] M. Harman, L. Hu, X. Zhang, and M. Munro, "Side-Effect Removal Transformation," Proc. IEEE Int'l Workshop Program Comprehension (IWPC 2001), pp. 309-319, May 2001, available at http://www.brunel.ac.uk/~csstmmh2/linsert/.

[7] D. Hendrix, J.H. Cross II, and S. Maghsoodloo, "The Efectiveness of Control Structure Diagrams in Source Code Comprehension Activities," IEEE Trans. Software Eng., vol. 28, no. 5, pp. 463-477, May 2002.

[8] M. Hills and P. Armitage, "The Two-Period Cross-Over Clinical Trial," British J. Clinical Pharmacology, vol. 8, pp. 7-20, 1979.

[9] ISO, International Standards Organisation: Programming Languages—C, International standard, ISO/IEC 9899: 1990 (E), Dec. 1990.

[10] B.W. Kernighan and R. Pike, The Practice of Programming. Addison-Wesley, 1999.

[11] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El-Emam, and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," IEEE Trans. Software Eng., vol. 28, no. 8, pp. 721-734, Aug. 2002.

[12] R.O. Kuehl, Design of Experiments: Statistical Principles of Research Design and Analysis, second ed. Duxbury Thomson Learning, 2000.

[13] R.V. Lenth, "Some Practical Guidelines for Effective Sample Size Determination," The Am. Statistician, vol. 55, no. 3, pp. 187-193, 2001.

[14] L.F. Marshall and J. Webber, "Gotos Considered Harmful and other Programmers' Taboos," Technical Report 699, Dept. of Computing Science, Univ. of Newcastle, 2000.

[15] R.J. Miara, J.A. Musselman, J.A. Navarro, and B. Shneiderman, "Program Indentation and Comprehensibility," Comm. ACM, vol. 1983, no. 11, pp. 861-867, Nov. 1983.

[16] S. McConnell, "Keep It Simple," IEEE Software, vol. 13, no. 6, pp. 127-128, Nov. 1996.

[17] J. Miller, J. Daly, M. Wood, M. Roper, and A. Brooks, "Statistical Power and Its Subcomponents—Missing and Misunderstood Concepts in Empirical Software Engineering Research," Information and Software Technology, vol. 39, no. 4, pp. 285-295, 1997.

[18] G.A. Milliken and D.E. Johnson, Analysis of Messy Data, volume 1: Designed Experiments. Cambridge Univ. Press, 1984.

[19] P.W. Oman and C.R. Cook, "Typographic Style Is More than Cosmetic," Comm. ACM, vol. 33, no. 5, pp. 506-520, May 1990.

[20] J. Stevens, Applied Multivariate Statistics for the Social Sciences, third ed. Lawrence Erlbaum Assoc., 1996.

[21] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering: An Introduction. Kluwer Academic, 2000.

[22] L.Q. Yue and P. Roach, "A Note on the Sample Size Determination in Two-Period Repeated Measurements Crossover Design with Application to Clinical Trials," J. Biopharmaceutical Statistics, vol. 8, no. 4, pp. 577-584, 1998.

▷ **For more information on this or any computing topic, please visit our Digital Library at** http://computer.org/publications/dlib.