ABSTRACT

This paper describes two experiments that examined overconfidence in spreadsheet development. Overconfidence has been seen widely in spreadsheet development and could account for the rarity of testing by end-user spreadsheet developers. The first experiment studied a new way of measuring overconfidence. It demonstrated that overconfidence really is strong among spreadsheet developers. The second experiment attempted to reduce overconfidence by telling subjects in the treatment group the percentage of students who made errors on the task in the past. This warning did reduce overconfidence, and it reduced errors somewhat, although not enough to make spreadsheet development safe.

Keywords: end-user computing; error; overconfidence; office systems; program correctness; spreadsheet; testing; user-developed applications

INTRODUCTION

Spreadsheet development was one of the earliest end-user applications, along with word processing. Spreadsheet development continues to be among the most widely used computer applications in organizations (United States Bureau of the Census, 2003). Although many spreadsheets are small and simple throwaway calculations, surveys have shown that many spreadsheets are quite large (Cale, 1994; Cragg & King, 1993; Floyd, Walls, & Marr, 1995; Hall, 1996), complex (Hall, 1996), and very important to the firm (Chan & Storey, 1996; Gable, Yap, & Eng, 1991).

Unfortunately, there is growing evidence that inaccurate spreadsheets are commonplace. For instance, Table 1 shows that recent audits of 88 real-
world spreadsheets have found errors in 94%; yet several studies only reported spreadsheets with serious errors. The implications of this ubiquity of errors are sobering.

As Table 1 shows, the field audits that measured the frequency of errors on a per-formula basis found an average formula error rate of 5.2%. This formula error rate explains why so many of the examined spreadsheets contained errors. Most large spreadsheets contain hundreds or thousands of formulas. Given the cell error rates found in field audits, the question is not whether large spreadsheets contain errors, but rather how many errors they contain and how serious these errors are.

These field audits and the experiments described later found three types of errors.

- Mechanical errors are mental/motor skill slips, such as typing the wrong number or pointing to the wrong cell when entering a formula.
- Logic errors are incorrect formulas caused by having the wrong algorithm or expressing the algorithm incorrectly.
- Finally, omission errors occur when the developer leaves something out of the model.

Although observed spreadsheet error rates are troubling, they should not be surprising. Human error research has shown consistently that for nontrivial cognitive actions, undetected and therefore uncorrected errors are always present in a few percent of all cognitive tasks (panko.cba.hawaii.edu/HumanErr). In software development, for instance, over 20 field studies have shown that about 2% to 5% of all lines of code will always be incorrect, even after a module is carefully developed (panko.cba.hawaii.edu/HumanErr/ProgNorm.htm).

In the face of such high error rates, software development projects usually devote about a third of their effort to postdevelopment error correction (Grady, 1995; Jones, 1998). Even after several rounds of postdevelopment testing, errors remain in 0.1% to 0.4% of all lines of code (panko.cba.hawaii.edu/HumanErr/ProgLate.htm).

The testing picture in spreadsheet development, however, is very different. Organizations rarely mandate that spreadsheets and other end-user applications be tested after development (Cale, 1994; Cragg & King, 1993; Floyd, Walls, & Marr, 1995; Galletta & Huftnagel, 1992; Hall, 1996; Speier & Brown, 1996), and individual developers rarely engage in systematic testing on their own spreadsheets after development (Cragg & King, 1993; Davies & Ikin, 1987; Hall, 1996; Schultheis & Sumner, 1994).

In the face of large error rates in spreadsheet development and other human cognitive domains, why is testing so rare in spreadsheet development? The answer may be that spreadsheet developers are overconfident of the accuracy of their spreadsheets. If they think that
Accurately Determining Self-Efficacy for Computer Application Domains: An Empirical Comparison of Two Methodologies
www.igi-global.com/article/accurately-determining-self-efficacy-computer/37218?camid=4v1a