Error Trapping and Metamorphic Testing for Spreadsheet Failure Detection

Pak-Lok Poon, RMIT University, Melbourne, Australia
Huai Liu, RMIT University, Melbourne, Australia
Tsong Yueh Chen, Swinburne University of Technology, Melbourne, Australia

ABSTRACT
This study deepens the research on error trapping (ET) and metamorphic testing (MT) for detecting spreadsheet failures. It is observed that some spreadsheet developers and testers are confused between ET and MT, because the two techniques are similar to each other in some aspects. Inspired by the observation, this paper first outlines the main concepts of ET and MT using several examples for illustration. This is followed by discussing an experiment with a view to investigating and comparing the failure detection capabilities of the two techniques. The results of the experiment indicate that neither technique is sufficient in spreadsheet testing. Rather, ET and MT are complementary and they should be used together in spreadsheet testing whenever possible.

KEYWORDS
Failure Detection, Metamorphic Relation, Oracle Problem, Spreadsheet Testing, Test Oracle

INTRODUCTION
Nowadays, spreadsheet-based systems (or simply spreadsheets) have become an indispensable tool in various business areas such as accounting and financial reporting, asset recording, production scheduling, and engineering design (Lu et al., 1991; McDaid & Rust, 2009). Spreadsheets are often used to generate important information for managers and executives in making strategic decisions (Caulkins et al., 2007; Kruck et al., 2003).

Although spreadsheet applications are extremely popular, numerous spreadsheet failures [1] have been reported in the literature (Bishop & McDaid, 2011; Leon et al., 2015; Morrison et al., 2002; Panko, 1998, 1999, 2007; Panko & Aurigemma, 2010). Some studies further substantiate the widespread growth of faulty spreadsheets. For example, Panko & Ordway (2005) found that nearly all (94%) of the spreadsheets contained faults. A possible reason contributing to the large number of faulty spreadsheets is that their development has shifted from IT professionals to “non-technical” departmental end users such as accounting or marketing staff (hereafter referred to as end-user programmers); many of the latter have little formal training in software development and testing (Powell et al., 2008, 2009). According to the European Spreadsheet Risks Interest Group (EuSpRIG), faulty spreadsheets could result in various business risks including: (a) loss in revenue, profit, cash, assets, and tax; (b) mispricing and poor decision-making; and (c) financial failure.

In view of the above risks, several systematic techniques have been developed for dynamic testing (that is, testing involving software execution) of spreadsheets. These techniques include the constraint-based spreadsheet testing method (Abraham & Erwig, 2006), the “What You See Is What You Test (WYSIWYT)” methodology (Fisher et al., 2006), error trapping (ET) (Jain, 2010;
Powell & Baker, 2009, pp.115−116), and metamorphic testing (MT) (Chen et al., 2003; Liu et al., 2014; Poon et al., 2014).

Among these techniques, the constraint-based spreadsheet testing method requires formal training in software engineering, and the WYSIWYT methodology requires some technical knowledge of data-flow adequacy criteria (Jee et al., 2009) and coverage monitoring (Vilkomir et al., 2003). These requirements pose difficulties to end-user programmers who often do not possess such technical knowledge. On the other hand, ET and MT are more applicable to end-user programmers because both require less technical knowledge for their use [2]. Since this paper is related to dynamic spreadsheet testing from the end-user programmer’s perspective, ET and MT are the main focus.

In the course of our research into spreadsheet testing, we observe that many people confuse between ET and MT, because they are similar in some aspects. Thus, this paper aims to help readers understand the two concepts. It also investigates their failure detection effectiveness, and determines whether they are complements or substitutes. The paper first analytically compares the two techniques, specifically focusing on their similarities, differences, and relationship. The analytical comparison is followed by a discussion of an experiment, which investigates their effectiveness in detecting spreadsheet failures. Thereafter, a practical guidance on spreadsheet testing with the use of ET and MT are provided.

**DYNAMIC AND STATIC TESTING**

Testing is a verification and validation technique, and is often categorized as being either dynamic and static. Dynamic testing involves executing the software system with test data, and then checking the output and the operational behavior of the software (Sommerville, 2011). In the checking process, if the actual output (or operational behavior) is found to be different from the expected output (or operational behavior), a failure is revealed, indicating that one or more faults exist in the software. Static testing (also known as human testing), however, does not involve software execution. Reviews, inspections, and audits are examples of static testing (Myers, 2004).

In organizations, program code is often tested in the preliminary construction and final construction phases of systems development (Everett & McLeod, 2007, pp. 35−37). In the preliminary construction phase, code is static tested (for example, in the forms of code inspections or code walkthroughs). Myers (2004, p. 21) argues that, although not all testers read code, code review (a form of static testing) is widely accepted because this practice is quite effective in finding errors [3]. Thus, static testing should be performed between the time the program is coded and the time when dynamic testing commences. Dynamic testing is performed in both the preliminary construction and final construction phases in various forms such as function, performance, and load tests (Everett & McLeod, 2007, pp. 35−37). (Our paper focuses on functional testing, which attempts to identify discrepancies between the software system and its expected behavior from the end-user’s point of view (Myers, 2004, p. 129.) Although organizations typically spend about 25−50% of the total software development cost on testing (Watkins & Mills, 2011, p. 43), they rarely mandate that spreadsheets be tested after development (Panko, 2006b).

**ET**

Despite best efforts, wrong input data and formula faults may occasionally creep into a spreadsheet. These “upstream” problems may cause subsequent problems in “downstream” cells that depend on the “upstream” input data or computation results. ET could be used to detect these problems, and to minimize their adverse impacts on the spreadsheet users. More specifically, ET allows the end-user programmers to determine what happens in the event of an unintended run-time error, to prevent loss of recent changes to a spreadsheet, and to prevent the spreadsheet from refusing to function at all.

ET often takes the form of conditional-based tasks using built-in commands such as IF, IFERROR, ISERROR, and ISNA provided by the spreadsheet, or the ON ERROR command provided by Microsoft
Balancing Local Knowledge Within Global Organizations Through Computer-Based Systems: An Activity Theory Approach

Structured Techniques for Successful End User Spreadsheets