Abstract

Spreadsheets are one of the most common end-user programming environments in use today, and their importance in the business world continues to grow. Although considered critical software tools in many industries, particularly the financial sector, it is now widely accepted that the reliability of end-user developed spreadsheets is very poor. In order to address this problem, recent research has focused on the design and development of tools to aid end-users in the testing and debugging of spreadsheets. But to design tools that complement end-users natural behaviour, a knowledge of their behaviour within the most common commercial spreadsheet environment (Excel) is required. This paper details the design and application of a novel, custom built, data acquisition tool, which can be used for the unobtrusive recording of end-users mouse, keyboard and Excel specific actions during the debugging of Excel spreadsheets. The data acquisition tool was utilised in experiments to investigate debugging behaviour, and consequently formed the basis for the development and evaluation of a cell-coverage based debugging tool.

1. Introduction

The reliability of end-user developed spreadsheets has been shown to be poor following empirical and anecdotal evidence collected from studies investigating operational (real world) spreadsheet error rates, most of which are detailed in [8]. The most recent study that investigated spreadsheet error rates [9] reported that of 50 real-world operational spreadsheets audited, 94% contained errors. From the experience of a consulting firm, Coopers and Lybrand in England, 90% of spreadsheets with over 150 rows of data were found to contain one or more faults [8]. Due to the nature of spreadsheet use, it is the case that when failures do occur the results can be quite significant.

With regard to software tools that aid the spreadsheet debugging process, quite a few are available as both commercial and freeware applications, but in order to develop tools that complement end-user’s natural debugging behaviour, some understanding of that behaviour would be required. The concept of human-centered development in software engineering could be applied to the development of spreadsheet debugging tools. Norman [7] stated that:

“At its core, human-centered product development requires developers who understand people and the tasks they wish to achieve. It means starting by observing and working with users”.

Only a select few studies have observed and empirically recorded the behaviour of end-users while debugging spreadsheets, [4], [10] and [3]; studies in which participants behaviour was recorded using video taping and think-aloud protocols; data acquisition methods that are quite noticeable to the subject under observation. In order to unobtrusively analyse Excel end-user behaviour during the debugging process, a less visible data acquisition method would be required such as mouse-and-keystroke recording, but to date, very few spreadsheet debugging studies have done this.

The layout of this paper is as follows. Section 2 details the data acquisition methods that are most commonly used in human computer interaction (HCI) research, along with the data acquisition method chosen by the authors. In Section 3, the data acquisition method and tool designed by the authors is discussed and presented. In Section 4 the applications of the data acquisition method and tool in recording spreadsheet activity are detailed. A conclusion to the paper is presented in Section 5.
2. Data acquisition methods

There are a number of different methods for acquiring human computer interaction (HCI) data. These methods include video recording, screen recording, thinking-aloud protocol and eye-gaze tracking. All of these methods require a subject to be onsite with the researcher, or at the very least for the subject to install some recording software or equipment on their pc or within their working environment. These methods of data capture are somewhat intrusive by nature. A problem with data capture is that a person’s behaviour changes when they are aware that they are being observed; a phenomenon commonly referred to as the observer effect [1], and less commonly as the Panopticon effect or the Heisenberg effect [6].

With this in mind, the authors developed a mouse-and-keystroke recording tool, that could be embedded within an Excel spreadsheet, and as such be used remotely by any subject without the need for the author’s presence. One of the main advantages of recording mouse-and-keystroke induced UI (user interaction) events is that they “provide excellent data for quantitatively characterising on-line (on-screen) behaviour” [5], and with this type of non-intrusive system monitoring the “influence on participants by observation is zero” [11]. The authors believe that the influence on participants would not be quite zero, as subjects should be clearly informed of the data capture method, but nevertheless, the influence would be considerably lower than that of video recording and think-aloud methods. The next section details the design and workings of a data acquisition tool developed by the authors.

3. Data acquisition tool: T-CAT

A ‘time-stamped cell activity tracking tool’ (T-CAT) was developed by the authors in VBA, and makes use of Microsoft Excel’s macro programming environment. The main advantages of developing T-CAT in VBA were firstly that the tool could be embedded in experimental spreadsheet models with no software installation required by subjects (especially relevant for off-site participants), and secondly, the tool could easily access MS Excel’s event listeners such as the Workbook functions: Open, BeforeClose, SheetActivate, SheetChange, and SheetSelectionChange.

The tool was designed to record the time and detail of all cell selection and cell change actions of individuals when interacting with a spreadsheet. The T-CAT tool ‘listens’ for cell activity events, such as: worksheet selections, cell selections and cell edits. When an event occurs, the tool records all the details associated with the event, and stores them to arrays. When the spreadsheet is closed, MS Excel’s (VBA’s) BeforeClose event is initiated, and values stored in the arrays are printed to a hidden worksheet within the experimental spreadsheet.

As mentioned earlier, the data recorded by T-CAT is printed to a hidden worksheet within the spreadsheet when the spreadsheet is closed. Figure 1 shows a sample of debugging data captured by T-CAT during a spreadsheet debugging experiment carried out by the authors. Looking at cell ‘A3’ from Figure 1, the value shows that the cell ‘D10’ on the ‘Payroll’ worksheet was edited after 40.828125 seconds, with a resulting cell value of ‘40’. The process of editing this cell can be determined by looking at cells ‘F32’ and ‘F33’ in Figure 1; ‘Payroll D10’ was selected after 33.046875 seconds, was edited, and then the Return key was pressed bringing the focus to Cell D12. Cell ‘F33’ shows that the cell edit was completed after 40.828125 seconds elapsed time. This gives a time for editing cell ‘Payroll D10’ of 7.78125 seconds.

![Figure 1: Sample recorded data](image)

4. Application of T-CAT

4.1 Recording spreadsheet activity

The T-CAT data acquisition tool was an integral part of a study [2] undertaken by the authors that aimed to record and analyse the behaviour of 47 professional and student spreadsheet users while debugging a spreadsheet model seeded with errors. The T-CAT tool allowed for the recording of the following data: individual cells selected, cell ranges selected, worksheet selections, individual cells edited and the resulting cell or formula values, cell ranges changed...
and resulting cell or formula values. Timestamps in seconds and milliseconds were recorded for all of the listed actions. Based on analysis of the data recorded during the study, it was possible to accurately determine many debugging actions and behaviours. Some of these are listed below:

- Overall debugging time taken by participants
- Cell values edited/re-edited
- Order in which sheets and cells were inspected
- Number and addresses of cells inspected, and the exact time taken to inspect each cell
- Use of drag-and-fill for copying logical areas

4.2 Creating a debugging tool

From analysis carried out by the authors on experiment results from [2], a moderate to strong correlation was found to exist between the number of cells inspected and debugging performance. Based on this finding, a simple spreadsheet debugging tool was developed. The debugging tool runs concurrently with the T-CAT tool, and uses data gathered by T-CAT to give feedback to users on cells that have been inspected during the debugging process.

The debugging tool is installed as an Excel add-in. Once installed, a button titled ‘Highlight’ is added to each worksheet; see Figure 2. When a user clicks on this button, any cells that have not yet been edited, or selected for a minimum specified time of 0.3 seconds, become highlighted. The user can then inspect the highlighted cells. The minimum specified time of 0.3 seconds was set based on analysis of the minimum time it took participants to inspect a cell. If the Highlight button is clicked again, the highlighting is updated. An example of a worksheet in which the Highlight button has been clicked, with any cells that had not been inspected highlighted in grey, can also be seen in Figure 2.

4.3 Evaluating the debugging tool

In order to determine the effectiveness of the debugging tool, an experiment was conducted with 16 fourth year Software Development students. The eight participants in the control group, Group-A, debugged a spreadsheet seeded with errors without the debugging tool. The cell coverage tool was made available to the 8 participants in the test group, Group-B, and instructions were given to them on how the tool worked.

The T-CAT data acquisition tool, of which all the participants were made aware, allowed for analysis to be carried out after the experiment was conducted. The test group using the cell coverage tool were found to have achieved a significantly higher cell coverage rate than the control group, with 90% and 26% coverage respectively. The test group corrected only slightly more errors, 62% vs. 59%, a result which is not statistically significant.

Although the 8 fourth year Software Development students in Group-B were given the same instructions, there were some differences in the way each student used the debugging tool. The scope and exactness of the data recorded through T-Cat tool supports the detailed examination of the impact of the debugging tool on behaviour.

Figure 3 shows a debugging behaviour that appears to have been influenced by the use of the debugging tool. When the highlight button on the ‘Payroll’ sheet was clicked, 18 cells became highlighted. This process was repeated for the second and third sheets of the spreadsheet, with 5 and 0 cells being highlighted respectively. As the participant debugged each sheet and used the cell coverage tool, a conscious effort appears to have been made to inspect all the cells.
The behaviour of two other participants, as represented in Figures 4 and 5, involved the use of the cell coverage debugging tool to a greater extent than the participant in the previous example. Figure 4 shows that the highlight button on the ‘Payroll’ sheet was clicked four times. It also shows that the participant re-inspected the ‘Payroll’ and ‘Office Expenses’ sheets, using the coverage tool each time. Overall, analysis of the data collected by T-CAT showed that subjects interacted with the debugging tool in many different ways and to different extents.

![Figure 4: Debugging tool - user behaviour (2)](image)

![Figure 5: Debugging tool - user behaviour (3)](image)

5. Conclusions

This paper details the successful implementation of a data acquisition method and tool for the spreadsheet research area. Although having advantages in its own right, namely the unobtrusive capturing of subjects actions, the mouse-and-keystroke recording method does lack the qualitative aspects of video capture and think-aloud protocols. Future work could investigate a combination of these data-acquisition approaches, where the aim would be minimal influence upon the behaviour of subjects being observed, while still gathering rich quantitative and qualitative data to aid and enhance the development of spreadsheet tools.

6. References


