An Exploratory Study of Spreadsheet Debugging Processes

Ying Chen
National University of Singapore

Hock Chan
National University of Singapore

Follow this and additional works at: http://aisel.aisnet.org/pacis2000

Recommended Citation
http://aisel.aisnet.org/pacis2000/12

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2000 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
An Exploratory Study of Spreadsheet Debugging Processes

Ying Chen, Hock-Chuan Chan
School of Computing
National University of Singapore

Abstract
In order to provide cognitive support for spreadsheet debugging, debugging processes of users without spreadsheet and accounting expertise were examined in terms of a model of spreadsheet debugging. Four participants were trained and asked to debug spreadsheets while thinking aloud. Participants’ frequent use of new tools (such as the auditing tools in Excel) to display cell relationships showed that they had considerable difficulty in reconciling surface and deep structures in spreadsheets. Furthermore, they could not identify the overall logical flow when they explored the spreadsheets. Hence, a frequent debugging strategy is a simple flow of attention from top to bottom and left to right. Data organized in the same way may facilitate users’ debugging. Finally, the factors of ease of use and perceived usefulness were found to have effects on the selection of debugging operators.

Keywords: spreadsheet, error detection, surface structure and deep structure

1. Introduction

Spreadsheets are applied to a wide variety of tasks, such as forecasting, trend analysis, “what if” analysis, goal seeking and model optimisation. Three main reasons contribute to its popularity. Firstly, spreadsheets combine an expressive high-level formula language with a powerful visual format to organize and display data so that they are easy to learn (Baxter and Oatley, 1991) and their professional appearances increase their credibility. Secondly, managers can use spreadsheets effectively to guide their strategic decisions (Hall, 1996). Thirdly, managers can gain independence from central computing controls. This means that, generally, managers can fulfil their task without help of professionals and do not need to learn programming.

Although spreadsheets are powerful and easy to use, simple and yet serious user errors can easily occur. Several reports illustrate this situation (Panko, 1994; Judith, 1996; Ditlea, 1987; Creeth, 1985; Dhebar, 1993; Freeman, 1984; Godfrey, 1995; Freeman 1996). For example, a consulting firm reported that over 90% of all spreadsheets with more than 150 rows contained at least one significant formula mistake (Freeman, 1996).

Several studies have been done to investigate manual error detection with different traditional methods, such as on-screen display of formulae, or printed versions of the spreadsheet and/or formulae. However, improvements measured in these studies are not ideal.

1 Ying Chen is currently with National university of Singapore, Singapore where she is enrolled for her doctorate degree program and can be approached by email: chenying@comp.nus.edu.sg. The authors gratefully acknowledge the many useful suggestions and contributions provided by reviewers.

2 Dr. Hock-chuan Chan is associate professor at school of computing.
The understanding of cognitive activities during spreadsheet debugging may be important for identifying the critical factors influencing error detection performance, and thereby to derive new effective and efficient debugging strategies and tools.

In this study, a problem-solving framework is adopted to examine different cognitive debugging processes, which are integrated into a debugging model. The impact of surface and deep structures in spreadsheets is also examined. Due to the different characteristic of debugging aids, it will be very helpful to determine whether ease of use or perceived usefulness of an operator has any effect on its frequency of use. The later sections are organized as follows: two reasons for making errors are reviewed in Section 2; a model of spreadsheet debugging is explained in Section 3; debugging operators are described in Section 4; research method is explained in Section 5; result and discussion are shown in Section 6; and a conclusion is given in Section 7.

2. Reasons for Spreadsheet Errors

This section reviews two major reasons for errors in spreadsheets.

2.1 Discrepancy Between Surface and Deep Structures

Generally, two levels of structure exist in a spreadsheet: visual/surface structure and computational/deep structure. The structure can be described at two different granularities: Cell and Module. From cell perspective, spreadsheets are viewed at micro level. Elements are cells while a structure represents the relationships among them. In contrast, module is used to describe spreadsheets at macro level. A module is made up of a set of cells that have the same function in a spreadsheet model. The structure describes relationships among modules. It is an overview of how a spreadsheet operates.

Often, the surface structure of spreadsheets is inconsistent with the deep structure at both cell and module level. At cell level, the inconsistency can be expressed in two ways (Saariluoma and Sajaniemi, 1989):

1. References in a single formula is non-unitary and non-organized, as in E6 * F9 * A5 * C4, letters and figures are not listed orderly;
2. References in several formulae are interwoven in “the cell dependency network”. For example, if the formula in cell B1 is A1 + A4 * A5 while the one in cell B3 is A2 * A3 + A6, then links between references and formulae are crossed (see figure 1).

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>A1 + A4 * A5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>A2 * A3 + A6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 1. Surface and deep structures in a spreadsheet**

The situation at module level is similar to that at cell level. So it is difficult for users to understand calculations in spreadsheets when two level structures are very inconsistent, because information search is controlled by the surface structure. The memory load that is
used to extract the computational structure increases largely (Saariluoma and Sajaniemi, 1991). As a consequence, users will have to spend more time on learning and memorizing the deep structure to do spreadsheet calculations.

This suggests that direct spatial mapping between surface and deep structures could be an effective design strategy to alleviate memory load. Formulae with similar functions should be put in neighbouring places.

2.2 Lack of Systematic Debugging

In programming, the debugging stage may take up to a third of total development time and effort. Measured error rate for a program is about 3%-7% in terms of all lines of code (Beizer, 1990). However, for spreadsheet development, systematic debugging is seldom included. Spreadsheet developers do not spend much time on systematic debugging (Brown and Gould, 1987). Two reasons might explain this phenomenon:

1. Compared to programs, spreadsheet models have fewer opportunities for reuse. Standards for spreadsheet models are only put into operation very locally.
2. Though some researchers are aware of many errors contained in spreadsheets and suggest that a systematic debugging stage should be required, few companies have built corresponding rules. When asked requirements for post-development audits in a survey of 107 MIS executives using a mail questionnaire, only 15% said that they had rules, and only 34% said that they had guidelines (Galletta and Hufnagel, 1992).

Therefore, it is necessary to develop new debugging tools to facilitate spreadsheet debugging. The examination of cognitive activities during spreadsheet debugging may give insights to develop cognition supported debugging tools.

3. A Model of Spreadsheet Debugging

Currently, three dominant methods are used for error detection: presentation (Galletta et al., 1997), code inspection (Panko, 1997) and interactive tools (Davis, 1996). There is little research to examine cognitive aspects of spreadsheet debugging. However, users’ cognitive activities play an important role in implementing their task. Hence, it is critical to gain an overall understanding of cognitive activities so as to identify the critical components of debugging processes. In addition, some tools can improve error-finding rate by making “hidden” errors more obvious or by reducing the memory load. Therefore, the second objective is to investigate features of selected debugging operators. Operators are described in the next section.

When users without spreadsheet and accounting expertise try to discover errors in spreadsheets, they will go through several essential steps to achieve their goal. The main goal of error detection is to find errors. So the first step of users is to read the cells and understand the relationships among cells. Sometimes users could divide the cells into different logical chunks and check them chunk by chunk. Hence, there are two levels of understanding spreadsheets: at cell and module level. Then the second step is to find problems related to the spreadsheets. Once the problem is identified, users will form their hypothesis about why and how the error is made. The last step is to locate and confirm the error in terms of the hypothesis. If the hypothesis is correct, an error is found. Otherwise, repeat the steps mentioned above. The correspondingly error detection procedure in spreadsheets based on debugging program functions (Vessey, 1985) is shown in figure 2.
Due to the similarity of cognitive activities in debugging programs and spreadsheets, the essential episodes derived by Vessey (1985) could be applied to debugging processes in spreadsheets. The model needs to be empirically tested.

Essential episodes to detect errors in spreadsheets are listed as follows:
1. Determine problem;
   Compare expected results with actual values. This is the basis to form a hypothesis of what an error could be.
2. Gain familiarity with spreadsheet (at cell level);
   Users try to understand relationships among cells. What they do is to read the content in spreadsheets and find out cell dependencies. They may read cells row by row or column by column.
3. Explore the structure of spreadsheet model (at module level);
   Users try to understand components that make up spreadsheets and their relationships. The mental representation of spreadsheets at macro level is formed. Users may check a set of relevant cells in an area during a period of time repeatedly.
4. Repair error.
   After a problem is found, the tracing procedure begins. Due to cell dependencies behind the spreadsheet model, the error might be made in the problematic cell or its precedents. So check the problematic cell and its precedents to locate an error.

![Diagram of error detection in spreadsheets](image)

**Figure 2. Model of error detection in spreadsheets**

However, these episodes may occur in a non-linear way. Sometimes in order to identify a hidden error, a user might check the spreadsheet under different suppositions. Therefore, before determining a problem, users may explore the spreadsheet several times. And then these episodes may be created in different order so that they could have an effect on users’
work efficiency and effectiveness. Hence, it is important to verify the existence of these episodes and clarify their relationships.

4. Debugging Operators

Debugging tools are useful when users looked for errors in spreadsheets (Davis, 1996). According to their availability and the users’ preference, they can be applied to different situations. For example, users could just explore spreadsheets for errors when only neighbouring cells are affected by errors. However, if the relationships among cells are very complicated and related cells are scattered, to display a deep structure of the spreadsheet will alleviate users’ memory load and could be a better choice than to explore spreadsheets. The following sections will describe selected debugging operators.

4.1 Trace precedent (TP)

Trace precedent draws arrows to a selected cell from all cells that appear in its formula. It can be used to identify all cells influencing the selected cell.

4.2 Trace dependent (TD)

Trace dependent draws arrows from the selected cell to all cells that refer to it.

4.3 Highlight feature (HL)

It is activated by double clicking the selected cell. Its formula appears in the cell position, and color codes link the cell references to the actual cell locations. Highlight feature is similar to TP operator in showing precedents. TP can be used for multi-level tracing while HL cannot.

4.4 Auto-calculate feature (AC)

Auto-calculate feature is useful when a user wants to find the sum or the average over a range of cells. It is not able to check relations among cells.

4.5 Printed spreadsheet (PS)

Printed spreadsheet can be applied in certain situations such as when users want to identify the structure of a complicated spreadsheet that can not be covered by one or two screens. It could help navigation in the spreadsheet.

4.6 Examining spreadsheet (ES)

Examining spreadsheet is the easiest to use. However, it may not be useful because users are basically exploring the spreadsheet on screen.

4.7 Spreadsheet description (SD)

Spreadsheet description is the printed description of what the spreadsheet is supposed to do. It includes points noted when users read a spreadsheet. It could help users to check the validity of each point or verify the suspected cells.
5. Research Objective and Method

The research study has the main objective of understanding the spreadsheet debugging processes. Under this general objective, the more specific objectives include:

1. Explore the spreadsheet debugging model (figure 2),
2. Explore the importance of deep structures, and
3. Explore the effect of ease of use and perceived usefulness on the use of debugging operators.

The research method is based on in-depth analyses with video recording and protocol analysis. Verbal protocol analysis was used as a data collection method in this study. Four participants were selected. This number is sufficient, as Nielson, 1994, reported for a thinking aloud test, four to five subjects were enough to identify about 80% of the problems. Protocol analysis is a thinking aloud process for the participants. Additional subjects would give marginal new information. The participant in this study had to understand four spreadsheets in Excel 97 and then debug the errors. Post-questionnaire for ease of use (EU) and perceived usefulness (PU) filled by the participants after the experiments were used to analyse their preferences for debugging operators. EU and PU are the main inputs of the Technology Acceptance Model (TAM) and TAM is preferable when the central goal is to understand information technology usage (Taylor and Todd, 1995).

5.1 Procedure

The participants were four computer science undergraduates. Hence, they were assumed to have lots of computing experience, and would be able to learn to use the debugging operators quickly. They were given the training material (Working with Excel) and the worked example (Get Rich Quick – Financial Forecast) at least one day in advance. They were encouraged to go through the material given but it was not compulsory. How much they understood was tested by their working through the worked example under the supervision of the researcher. The researcher would make sure that participants could use the various type of debugging operators before they started to debug.

The study of spreadsheet debugging was divided into four sessions. One spreadsheet was assigned to one session. Four spreadsheets, “Annual Student Budget”, “Monthly Expense of ABC Building Company”, “Bid a Wall” and “Final Results”, were presented to participants in sequence. Each session was given 45 minutes, with 5 to 10-minute breaks between sessions. Description of the spreadsheet, printed spreadsheet and an answer sheet were attached to each spreadsheet. The answer sheet was filled by participants to record starting time and ending time of debugging, current time to find errors, corrected form of errors, how they manage to find errors and percentage of errors they think have been found. The whole debugging process was videotaped and the screen activities of participants were recorded. To better coordinate the data collected by the video camera and screen capture, the researcher would do some manual recording. At the same time, she would prompt the participants to “think aloud”. When debugging was finished, participants filled in an exit questionnaire. This was to gather information of their background and their perception towards the debugging study.
5.2 Spreadsheets

The task of a participant was to find errors planted in the four Excel spreadsheets. Error distribution of spreadsheets is summarized in Table 1.

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Errors</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annual Student Budget</td>
<td>The annual budget of a student based on his or her income, school fees and living expenses for two semesters</td>
<td>9</td>
<td>Galletta et al. (1996)</td>
</tr>
<tr>
<td>Monthly Expense of ABC Building Company</td>
<td>Overhead, material and labour expenses spent by a company each month</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Bid a Wall</td>
<td>The bid to build a wall using two kinds of brick and concrete</td>
<td>4</td>
<td>Panko et al. (1996)</td>
</tr>
<tr>
<td>Final Results</td>
<td>Final results of students, based on performance in their homework, quizzes and final exam</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

6. Results and Discussion

6.1 Debugging Process

Table 2A presents the sequence of explored cells, applied debugging operators, correspondent episodes, time for reading each description and duration of discovering each error for participant 1 on “Annual Student Budget”. Table 2B shows the same information about participant 1 with “Monthly Expenses of ABC company”. Table 2C shows the information for participant 2 on “Monthly Expenses of ABC company”. Other participant-spreadsheet combinations are considered, but not included here. The boldface represents the cells with error and the word before cells is the abbreviation of the debugging operator used by participants. Edit(cell) indicates that participants edited the cell while restore(cell) shows that the cell was restored to its original content.

According to the spreadsheet debugging model, four episodes are critical components of the debugging process. They can be identified through combination of the sequence of debugging operators used and verbal recording of participants. Debugging operators alone cannot be used to identify the episodes because some of them can be applied to more than one episode. For example, trace precedents (TP) can be used to understand the structure of a spreadsheet when a user does not have any impression about an error while it can be used to locate errors once the user has discovered a suspicious cell. However, the recording of their use can help us to confirm the occurrences of these episodes. Repetitive browsing of the same cell indicates that participants wanted to locate and confirm errors while the use of debugging operators on special cells is the beginning of checking errors.

The sequence of episodes listed in Table 2A to Table 2C is irregular. It is consistent with the model. It is found that participants explored the spreadsheets by stages. They read through the descriptions before they started debugging. They would stop debugging and check the descriptions when they could not understand the contents in the cells or when they needed additional information. Then they began the next exploration. Participants spent less time on repairing errors. Participants could identify errors quickly if they found problematic cells. It indicates that more efforts should be put on how to help users understand and identify spreadsheet models quickly.
Two distinct strategies during spreadsheet debugging were identified. One strategy is a simple top-down and left-right scan. Three participants used this. Participants began to check each spreadsheet after spending a few minutes, about 2 to 5 minutes, reading its description. They did not think about the overall spreadsheet structure in advance. This phenomenon is similar to that of spreadsheet creation stage by Brown and Gould (1987). Hence, the first thing they did was to understand the spreadsheets cell by cell. They read the spreadsheet bit by bit while correcting the errors planted in them. Cells were explored from the top-left to bottom-right. However, sometimes they could identify a part of the spreadsheet structure and would make use of it for error detection. For example, when participant 1 examined “Monthly Expenses of ABC company”, she recognized that column B is SUM of columns C to F and row 14 is SUM of rows 8 to 13 by checking the precedents of cell C9 to F9. After that, she just quickly validated B9 to B13 and C14 to F14 to find the error in F14 without checking the precedents of C9 to F13.

| Table 2A. Cognitive activities of participant 1 and spreadsheet 1 |
|---|---|---|---|
| Title | Episodes | Sequence | Duration (min) /Distance |
| Annual Student Budget | Reading description | N/A | 2 |
| | Gain familiarity | ES – TP(E4) – SD – TP(E6) – TP (F6) – ES | 6 |
| | Determine problem | TP(D7), AC(D14, D24) | |
| | Determine problem | TP(E7), AC(E14, E24) | |
| | Repair error | edit(E7) – restore(E7) – edit(E14) – TP, TP<sup>edm</sup> (E7) – restore(E14) – edit(E7) – TP (E7), AC(E14, E24) | 15/60 |
| | Gain familiarity | SD – TP(F7) – ES – TP(D10) – ES, SD – TP(E10) – TP(F10) – TP(E11) – ES –TP(D14) – ES(E14:F17) – TP(E19) | |
| | Determine problem | AC (D19, E19) | |
| | Gain familiarity | TP(D20) – TP(E20) – TP(E21) | |
| | Determine problem | AC(D21, E21) | |
| | Explore structure | TP(D22) – TP(E22) – TP(D23) – TP(E23) – TP(D24) – TP (E24) | |
| | Determine problem | TP(F24), AC(D24, E24) | |
| | Repair error | Edit(F24) | |
| | Gain familiarity | ES | 2 |

| Table 2B. Cognitive activities of participant 1 and spreadsheet 2 |
|---|---|---|---|
| Title | Episodes | Sequence | Duration (min) /Distance |
| Monthly Expenses of ABC company | Reading description | N/A | 5 |
| | Gain familiarity | ES – AC(B4) – ES(B4:F4), B5:F5 | 2 |
| | Determine problem | ES(B8:B13) – SD(B11) | |
| | Repair error | edit(B11) | |
| | Gain familiarity | ES(B8:B13) | 10/30 |
| | Determine problem | AC(B8:B13), ES(B14) | |
| | Gain familiarity | ES(C8:F8) – TP(C9) – ES(C10) – TP(C11:C13) – ES(C8:C13) – TP(C9: F9) | |

150
<table>
<thead>
<tr>
<th>Title</th>
<th>Episodes</th>
<th>Sequence</th>
<th>Duration (min) /Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monthly Expenses of ABC company</td>
<td>Reading description</td>
<td>N/A</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Explore structure</td>
<td>SD – HL(B14:F14)</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Determine problem</td>
<td>HL(F14), ES(F14)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Repair error</td>
<td>edit(F14) – ES(F14:B14)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gain familiarity</td>
<td>HL(B5), ES(C5:F5) – ES, SD</td>
<td>3/12</td>
</tr>
<tr>
<td></td>
<td>Determine problem</td>
<td>SD(B11)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Repair error</td>
<td>edit(B11)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gain familiarity</td>
<td>SD, ES – HL(D9:F9) – HL(C11:F11) – HL(C12)</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Determine problem</td>
<td>ES(C12:F12)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Repair error</td>
<td>edit(C12:F12)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gain familiarity</td>
<td>HL(C13:F13)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2C. Cognitive activities of participant 2 and spreadsheet 2

The second strategy relies on some understanding of the spreadsheet’s deep structure. One participant used this. Participant 2, who spent about the same time on reading spreadsheet descriptions, started to debug from the output part of the spreadsheet, the last row (Table 2C). The difference might be due to the fact that participant 2 benefited from taking an accounting course before, and had knowledge of the importance of the bottom line in a budget statement. It indicated that she would have an initial impression on the spreadsheet structure and did not spend extra time on it. Hence, training and experience might help users understand the spreadsheet structure better and thus enable a more directed error search. Participant 2 did detect more errors than others.

Participants often checked the spreadsheet descriptions and used trace precedent (TP) or highlight feature (HL) to display the relationships among cells when they examined the spreadsheets. The frequent use of TP or HL confirms that users cannot recognise deep structure and visual aids do help to clarify the relationships among cells. The alternation from screen to paper may reduce the speed to finish task and lower the accuracy of error detection.

Another interesting finding is that the duration to discover errors seems to be somewhat related to the cell position (Table 2): the shorter the “exploration” distance (roughly calculated by counting the number of explored cells between the erroneous cells) between neighbouring detected errors, the less time were spent discovering errors. This finding is not very conclusive because the difficulty level of an error may have an effect on the time spent on finding it. This finding is probably also obvious: the more cells the participant explore, the more time is needed. However, it does suggest an important method to shorten the time to find errors, i.e. to reduce the number of wasted cell explorations. Participants appeared to care more about cell values rather than formulae. Hence, they used the calculator or Excel tools to calculate cell values according to the formulae. They did not realize that they should check the correctness of the formulae first and then cell values. They did not extract the logical flow from the spreadsheets. Formulae in a
spreadsheet are similar to the algorithms in a program. The key to build a correct spreadsheet model is to use proper formulae. So to check the correctness of formulae is a critical step to find errors.

![Diagram showing the contrast of debugging operators](image)

**Figure 3. Contrast of debugging operators**

### 6.2 Use of Debugging Operators

Figure 3 shows the participants’ ranking of debugging operators for ease of use and perceived usefulness. The smaller the rank number, the easier to use and the more the usefulness perceived. Circled regions show a high concentration of points. The groupings indicate that participants roughly agreed with each other on perceived usefulness of debugging operators. The order of debugging operators from the most perceived usefulness to the least is Auditing Tool (AT, inclusive of both TP and TD), auto-calculate (AC), and lastly, printed spreadsheet (PS) together with examine spreadsheet (ES). The position for highlight feature (HL) cannot be determined because its evaluations are too scattered.

With the same visual analysis, the overall ranking based on ease of use will be AT together with ES, followed by AC, and lastly by PS. HL again has a highly varied ranking among the participants.

Figure 3 indicates very strongly that the newer debugging tools (auditing tools, auto-calculate feature and highlight feature) are highly ranked according to perceived usefulness, compared to the traditional tools (printed spreadsheet and examine spreadsheet on screen). Even for ease of use, the newer tools are as easy as scanning the screen (which is what ES is). The traditional tool of printed spreadsheet is ranked at the bottom. It is worth noting that there is an exception. Participant 4 ranked printed spreadsheet as the best.
for both perceived usefulness and ease of use, better than all the on-screen operators. This strongly suggests that some individuals may prefer to work with papers rather than screens.

Analysis of the most frequently used debugging operators of each participant for each spreadsheet shows that the choice of operator is very strongly related to ease of use. All most frequently used operators except ES are ranked 1 or 2 by its user. This indicates that if an operator is considered hard to use or of little usefulness, it will not be frequently used. ES is frequently used despite its low perceived usefulness. This could be because it is a “natural” operator (scanning a window) shared for all window operations so that participants could use it without considering its PU. It should be emphasized that the frequent use of TP, AC and HL indicates that sometimes participants did need to clarify the deep structure hidden in the spreadsheets.

7. Conclusion

With the general objectives of understanding processes in spreadsheet debugging, this study has offered strong indications in many aspects. The main points are summarized below.

The spreadsheet-debugging model is proposed for describing and studying debugging processes. Four episodes specify possible cognitive activities of users. This model was tested with an empirical study based on verbal protocol analysis. The evidence verifies the existence of four episodes in the model. Their occurrences were a little haphazard, with no definite sequence. It might be due to the fact that sometimes users have flexible goal choices.

An important finding is that participants did not have any overall strategies for spreadsheet debugging. They did not examine the spreadsheets as a whole though they sometimes would divide the spreadsheets into logical chunks. They tried to understand spreadsheets while correcting them. This piecewise understanding may be a cause of low error correction rate. Sometimes participants did feel some errors had not been detected yet. However, they did not know which method they should adopt. So debugging tools that can help users construct the flow of calculation based on logical chunks extracted from spreadsheets may make users understand better and improve error detection rate.

The importance of deep structures appear frequently, whether is localized understanding of a small group of cells, or in trying to understand the overall structure and modules of the spreadsheet. Participants used all kinds of operators to try to ascertain the relationships among cells. All the current operators show only cell to cell relationships. They cannot display logical chunks or modules and the inter-module relationships. Understanding modules and their relationships is a difficult task recognized during spreadsheet debugging processes. This suggests that new and better operators should be developed.

Ease of use and perceived usefulness are found to have strong indications on the choice of operators. Participants seemed to prefer to use the tools that are easy to use. Similarly, they frequently used operators with the highest perceived usefulness, with the exception of ES, which is frequently used although its perceived usefulness is low. The new operators, particularly AT, received favorable rankings on EU and PU, compared to traditional operators (ES and PS). This strongly indicates the feasibility and user acceptance of new
operators. Participants could use them freely and quickly to identify the relationships among cells.

Based on the findings, several further research works can be made. Firstly, systematic debugging strategies should be further derived from past research to study their impact on error detection. Secondly, more tools are needed to help show deep structures, not only on cell relationships but also particularly on higher level module information. Thirdly, much more research in understanding debugging processes, development of new tools, and understanding individual differences are needed in order to improve error detection significantly.

Reference


