Uncovering Effects of Programming Paradigms: Errors in Two Spreadsheet Systems

Markku Tukiainen

University of Joensuu
Department of Computer Science
P.O. Box 111, FIN-80101 Joensuu, Finland
Markku.Tukiainen@Joensuu.FI

Keywords: POP-I.B. choice of language, POP-II.B. design, modification, POP-III.B. spreadsheets.

Abstract

Empirical studies of spreadsheet programming have commonly shown high over all error rates but much less attention has been paid to reasons for these errors. One often mentioned cause for errors is the low conceptual level of spreadsheet systems. By changing the conceptual level of spreadsheet system, we wanted to study whether this will produce different type of errors compared to traditional spreadsheet systems. In this paper we present an empirical study comparing the traditional spreadsheet calculation paradigm with the structured spreadsheet calculation paradigm that utilizes goals, plans and spreadsheet data structures in computation. The results show that the error behavior of novice spreadsheet users is systematically different between paradigms.

Introduction

Traditional spreadsheet calculation systems provide a fast method to express data and to make calculations, provided that there is some natural way to lay all the data on one or more two-dimensional sheets. Spreadsheet calculation is the most widely used end-user programming language. The success of spreadsheet calculation systems has been attributed to their ease of use, users can see what is to be done and do it (Kay 1984) and to the metaphorical and visual nature of the user interface (Norman 1986). Despite this ease of use (for more balanced view, see (Lewis & Olson 1987)), spreadsheet applications tend to have a lot of faults (Panko 1997): Even experienced users made errors in 44% of the cases reported by Brown and Gould (1987). Their study also showed that the errors were mostly made in entering calculations, not simply in typing constants: Seventy percent of reported errors were errors in formulae.

There are several reasons for errors in using the traditional spreadsheet systems. First, there are errors due to the characteristics of the spreadsheets systems themselves. For example, formulae are not visible all the time; they are represented linearly, whereas the spreadsheet itself is presented spatially, and the physical distance on the screen between a formula and the cells referred to by the formula is large. All these features hide the true structure of a spreadsheet from the user, leading to errors (Brown & Gould 1987). Second, there are errors due to the physical limits of current computer systems. For example, if the amount of data entered is substantial, the physical limits of a computer screen force the user to see only a part of the application at a time. Usually, the user divides logically grouped data into smaller physical clusters in order to see and understand the workings of the application.

Third, there are errors due to limitations of the current spreadsheet calculation model. For example, naming the cells is the only way to structure computations (Ronen, Palley, & Lucas 1989), the specification of absolute and relative cell referencing is vague (Sajaniemi 2000, Doyle 1990), and the connection between the computation and the data is established by referencing to locations in a sheet,

not the actual data structures in the model. These reasons for errors have been attributed to the low conceptual level of current spreadsheet systems (Lewis & Olson 1987; Ronen et al. 1989).

The low conceptual level of current spreadsheet systems manifests itself especially in the absence of the connection between the computation in-the-large and the data areas. The spreadsheet user enters formulae once and is supposed to remember to make changes to the referenced addresses if the input data area changes in size. For example, if a user creates a table of values and then writes an expression using the addresses of the corner cells in the table or a named reference to the table and later adds a new row (or column) to the end of the table, the calculation (i.e. the formula) is not updated automatically.

Traditional spreadsheet systems have not changed much computationally since VisiCalc in 1979. The complexity of the interface and the amount of new features like easy graphing and helping agents have increased, but the formula specification remains relatively unchanged. One example of change in specifying the calculation can be found in recent versions of Microsoft Excel: the AutoSum feature. The user selects an area and then presses the AutoSum button and gets, by default, columnwise sums. Of course, the user can point to a row and press AutoSum and get a rowwise sum and then copy it across the needed areas. But the problem mentioned above remains the same; if the user adds a new row of values to the area, the AutoSum-formula is not updated to reflect this change in data.

Although traditional spreadsheet systems offer no other data structuring tools than range naming the users of a spreadsheet system do not see spreadsheets as collections of individual cells, but rather as logical collections of cells (Saariluoma & Sajaniemi 1989). This has led to a new approach to spreadsheeting called the structured spreadsheet calculation (Hassinen, Sajaniemi, & Väisänen 1988). One of the main ideas of the structured spreadsheet calculation is to make these implicit structures explicit so that the users can refer to these structures as a whole and connect these structures to computation so that the changes in structures will be reflected automatically to the computation. By offering this kind of abstraction, the conceptual level of the structured spreadsheet calculation becomes higher than the traditional spreadsheet calculation.

We wanted to examine whether this change of the conceptual level will have an impact on the number or the type of errors novice spreadsheet users will produce. Especially, if the types of errors are different this will have important implications on the further development of spreadsheet calculation systems. In this paper we present an empirical study comparing the traditional spreadsheet calculation paradigm and the structured spreadsheet calculation paradigm. The study was conducted using Microsoft Excel (a traditional spreadsheet system) (Microsoft 1994) and Basset (a structured spreadsheet system) (Tukiainen 1996).

The purpose of the study is to compare two spreadsheet paradigms, traditional and structured, having different conceptual levels. The subjects are, however, using actual spreadsheet systems (Microsoft Excel and Basset) that represent these paradigms. Some of the errors that subjects will make may be due to the specific system features while others may be affected by the paradigm used. Therefore, a clear distinction must be made between errors evoked by problems inherent to the paradigm and errors evoked by problems inherent to the spreadsheet system.

Experiment

The traditional spreadsheet calculation paradigm and the structured spreadsheet calculation paradigm differ quite much in their concepts and approaches. This raises the following question:

> *Do novice programmers of the structured spreadsheet paradigm produce different kinds of errors than novice programmers of the traditional spreadsheet paradigm?*

In the present experiment, we chose Microsoft Excel to be a representative system of the traditional spreadsheet calculation paradigm and Basset to be a representative system of the structured spreadsheet calculation paradigm. Since there are no expert programmers in Basset and the structured

spreadsheet calculation paradigm, we chose to use novice spreadsheet programmers as subjects. We taught them four hours of spreadsheet calculation in general and then they practiced the use of one spreadsheet tool (Excel or Basset) for four hours. All subjects did the same tasks but using the tool assigned to the group. We measured the number of error-free and erroneous solutions and classified the errors. We were only interested in errors in calculations, not in layout or entering numbers and headings. The data collected consisted of the files containing the saved spreadsheets and the pre-test questionnaires of subject's background. We also videotaped a few random subjects.

Method

The subjects were first-year undergraduate students at the University of Joensuu majoring in various areas of study ranging from theology to mathematics. They participated in the experiment as a course requirement for an Introduction to Spreadsheets course. A total of 154 students participated in the experiment. They were randomly assigned to Basset (74 subjects) and Excel (80 subjects) groups.

The experimental materials consisted of four spreadsheet applications. The tasks were ordered into 24 task sets, which contained all of the possible orders of the tasks. The task sets were distributed randomly to the subjects. All subjects did all of the four tasks using either Excel or Basset. The application domain was simple business-type accounting. This was chosen because it is a well defined domain, students usually have a fair understanding of the domain, and it is easy to design simple problems in this domain. Also, spreadsheets are heavily used in business accounting. The tasks chosen for the experiment were quite elementary, but they are representative of real-life spreadsheet tasks.

We used two types of tasks: construction and modification tasks. In the construction tasks the subjects had to implement new spreadsheet applications, and in the modification tasks they were given existing spreadsheet applications to modify. The applications varied in size from 22 cells to 33 cells and the number of formulae in applications varied from 3 to 5. The tasks involved data entry for numbers and headings, composing formulae with relative and absolute cell referencing, copying formulae and saving the application. All task instructions contained the headings and the constant numbers needed in the spreadsheets.

The tasks were:

- *Task 1A and 1B: The budget task and updating the budget task.* Construction of a new application. The subjects were asked to create a budget for their income and expenditure. The income items (3) and expenditure items (4) were given. The solution was to first enter the data for the income and the expenditure. Then they had to calculate the total sums of the income and expenditure and to calculate the difference between these sums. At this point the instructions asked them to save the application (Task 1A). After saving, the instructions asked the subjects to modify the income by adding one item to the end of the income data (Task 1B).
- *Task 2.: The cumulative sum of credit units.* Construction of a new application. The subjects were asked to create an accounting application for their credit units using accumulation. The instruction showed an example application without the formulae. The subjects were instructed to create an application with the same information, no demands for spatial layout were made. The credit units for 6 semesters were given. The solution was to enter the data and to create a cumulative sum.
- *Task 3. Adding new items to existing data structures.* Modification of an existing application. The application had two input data areas, one consisting of prices for 3 products and the other containing the numbers of sold items for each of the products. The titles for the areas and the items were taken from real-life settings, i.e., the titles were Prices, Sold Items, Hammers, Saws and so on. The application calculated sales values for the products using pair-wise multiplication of the prices and the numbers of items sold. The subjects were asked to add two new products into the application. The products were to be located as the second and the last item of the products. The formulae had to be updated to reflect these changes.

- *Task 4. Adding new calculations to an existing computation.* Modification of an existing application. The application to modify was the same as in the task 3. Subjects were asked to add a computation to calculate the sales values in Euros in addition to Finnish marks. The instruction showed an example application with the results of the formulae. The subjects were instructed to create an application with the same information; no demands for spatial layout were made. The solution was to add a cell containing a constant value for the exchange rate (given in the instructions), and to calculate the sales in Euros using multiplication of the sales in Finnish marks and the exchange rate.

The experiment took place in the same computer laboratories that the subjects had used for practicing sessions. The subjects were tested in groups of three to thirteen. The groups were the same as used in the practicing sessions. First the subjects were given a background questionnaire to answer. When all were ready, (in about five to ten minutes) the task sets with a blank page on the top were distributed to the subjects and oral instructions about the experiment were given.

The subjects were told that there was going to be five tasks similar to those they had done in the practicing sessions. The first task would be the same training task for all. Then the four experiment tasks would follow, being the same for all but in a random order. This was told so that the subjects would not wonder if the subject next to him would stop working earlier than himself. The subjects were told that there was a time limit of ten minutes per task. If they would complete the task before the time limit, they were instructed to open a new blank spreadsheet on the screen and wait until the instructor would give permission to start working on the next task. If they could not complete the task in the time limit, they were instructed to save the work as it was, with all the incomplete or erroneous calculations.

Results

The background of the subjects was controlled in the pre-test questionnaire where we asked their overall computer experience (variable A: ordinal scale), experience with spreadsheet calculation (B: years and C: number of applications done, ordinal scale) and experience in programming (D: years and E: size of largest application done, ordinal scale). Similarity of the groups was tested using Mann-Whitney Test (variables A, C and E) and t-test (variables B and D). There were no significant background differences between groups.

Figure 1 gives the overall results of task correctness. There are five tasks because the first task was divided into two subtasks, which were saved in separate files. The total number of correct sheets was 530 and the total number for incorrect sheets was 151. For various reasons 89 sheets were missing, the largest number of missing spreadsheets being with task 1 (total of 66 sheets missing). This task required the most input actions of all the tasks and it is reasonable to believe that the ten minutes time limit was not long enough for novice subjects and they did not save incomplete sheets. The single largest number of missing sheets was with Basset group in task 1B. Often the subjects did not finish the task 1A correctly, so they spend all the time doing that task. This happened in 20 cases of total 22 erroneous tasks 1A.

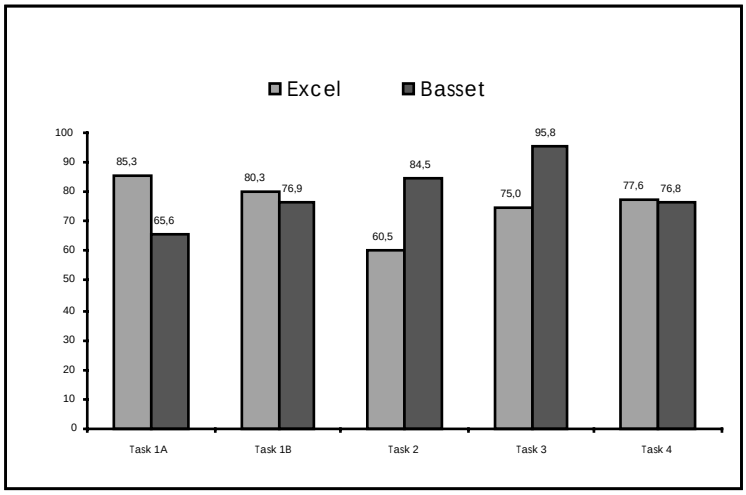

Figure 1 – Overall task correctness in percentages.

The Chi square test was used to test the interaction between the groups and the tasks. We tested the interaction between the task correctness (non-erroneous vs. erroneous sheets) and the tool used (Basset vs. Excel). Interactions were significant in three tasks: Task 1A ($\chi 2$=6.945, df=1, p=0.008), task 2 ($\chi 2$=10.496, df=1, p=0.001) and task 3 ($\chi 2$=12.449, df=1, p<0.001).

Error Analysis

We classified only errors that constituted distinctive groups. These groups are called error types (see Table 1). There were few errors that occurred only once or twice and these were not classified. Also logic errors, i.e., errors in understanding the task descriptions, were not classified. There were four researchers independently devising the causes for errors and all reasons are listed without taking up for consideration of their probabilities. Each error type is given a number starting with the letter E (Excel errors) or B (Basset errors).

We do not include in the possible error types any cognitive factors like losing attention during the task, because our experimental method was not planned for that type of observation and because such analysis is not relevant for the purposes of this experiment. We try to explain the errors in terms of the computation doctrine, i.e., spreadsheet-based computation (as opposed, e.g., to procedural programming), the spreadsheet calculation paradigm used and the spreadsheet calculation tool's usability issues (see Figure 2). Thus in the possible causes for each error type, classes doctrine, paradigm, and tool are used. In table 1, an error type name is followed by indications of possible causes for the error type. Each possible cause indication is given a label in which the first letter denotes the class (e.g. D = doctrine), and second letter denotes the tool (e.g. b = Basset). See Appendix A for longer descriptions.

Doctrine class includes error causes that could be due to the expected content or workings of the user's mental model of spreadsheet-based computation. This class contains also errors in mathematical concepts underlying the spreadsheet formulae. It has been noted that novice users' mental models in programming tend to be incomplete and erroneous (Bonar 1985). Doctrine class errors can arise in both spreadsheet calculation paradigms discussed in this paper.

Error type		Possible causes		
		Doctrine	Paradigm	Tool
E1	Sum includes empty cells at end		Pe1, Pe2, Pe3	Te1
E2	Not all appropriate cells included in sum		Pe4	
E3	An item added to both input structures		Pe5, Pe6, Pe7	
E4	Missing formula update		Pe8, Pe9	
E5	First item of cumulative sum is constant	De1, De2	Pe10	
E6	Malformed formula	De3, De4		
E7	Wrong reference in a new formula		Pe11, Pe12	
E8	Incomplete but correct computation		Pe13	
E9	Missing formula	De5	Pe14, Pe15	
E10	Malformed decimal number			Te2
B1	Missing goal	Db1	Pb1	Tb1
B2	Wrong goal applied		Pb2	Tb2
B3	Wrong arguments for a goal	Db2		Tb3, Tb4
B4	No item added into the input structure			Tb5, Tb6

Table 1 – The error types found.

Paradigm class includes error causes that could be brought about the reasons of spreadsheet calculation paradigm used. These errors can arise only in one of the paradigms discussed. This class is the most important for the current study because it reveals the error causes connected directly to calculation paradigms.

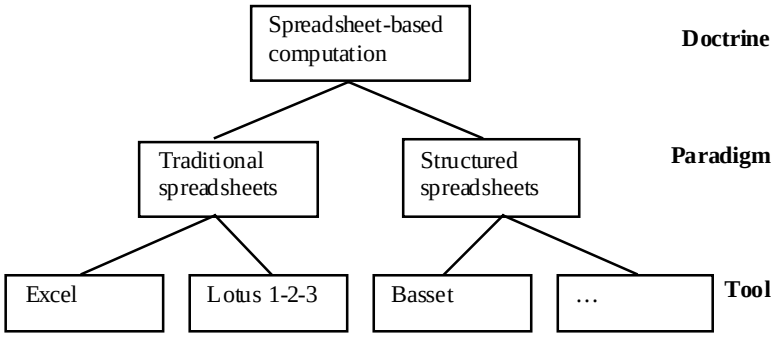

Figure 2 – The hierarchy of error origins.

Tool class includes error causes that could be brought about the usability problems in the specific spreadsheet tool used. This class is less important because we are not trying to study the usability of the tools but the differences in the paradigms behind the tools.

Discussion

The error analysis above defined the types of errors the subjects produced in each of the tasks using the two tools. There were total of 10 different Excel and 4 Basset error types. We will now discuss about the possible explanations behind these error types.

Doctrine Evoked Errors. Three Excel error types (E5, E6, E9) and two Basset error types (B1, B3) have possible causes that could be explained by the structure and content of the subjects' mental model of spreadsheet computation.

There seemed to be at least four types of possible reasons for subjects' doctrine evoked errors. First there were errors concerning misunderstanding of spreadsheet concepts like the difference between a reference to a cell containing a constant value and using a constant with the same value in computation. Second there were errors due to fragmentary knowledge of spreadsheet operations and usage. Examples of this could be when the subjects could not change the layout of the sheet to compensate the effects of inserting areas. Another example could be a misconception of Copy-operation. The same Copy-operation is used to copy constants and formulae and the subjects seemed to think that copying constant values is a kind of dynamic copy so that if the original constant value will change, the copy of that value will also change.

Third there were errors resulting from cognitively hard tasks, e.g., in the error type E6 subjects could not devise a plan for cumulative sum. Another example of this could be the errors concerning the identification of all correct and needed goals for the solution. The error type B1 (missing goals) could be explained by subject not succeeding in constructing the right goal for the task. Fourth there were errors in mathematical models underlying the spreadsheet implementations of formulae. These could be classified to a different (higher) class than spreadsheet-based computation doctrine because they could manifest themselves in other computation doctrines also.

Paradigm Evoked Errors. For the purposes of this paper, paradigm evoked errors are most interesting as they may reveal fundamental differences between structured and traditional spreadsheet paradigms. Eight Excel error types (E1, E2, E3, E4, E5, E7, E8, E9) and two Basset error types (B1, B2) have possible causes that may be explained by the paradigm specific features of the two tools used in the experiment.

The conceptual levels of the two paradigms differ. The paradigm errors we found in this study can be explained by this difference in conceptual levels. The Excel errors deal mostly with the low level details of formulae. This can be seen as a manifestation of the lack of data structuring tools and poor connection between data areas and calculations. On the other hand, the Basset errors occur at the data structure level of computation. We will first elaborate upon Excel errors and then upon Basset errors.

Typical errors in Excel are connected to copying. In the error type E1 the referenced area is too large and one probable cause is the misunderstanding of the Copy-operation. There are two input data areas, which are of different size. A subject adds empty cells to the smaller area probably because she may think that the referenced areas have to be of the same size in order for a Copy-operation to succeed.

Another misunderstanding of the Copy-operation can be found in error type E5. The same Copy-operation is used to copy constants and formulae, which may confuse a novice subject to over generalize the concept of Copy-operation. Furthermore, the traditional spreadsheet calculation paradigm does not separate constant value cells from formula cells in the cell surface appearance. This may make it hard for a novice subject to distinguish between the concepts of a copy of the cell content and a reference to the cell.

Changing the input data area, e.g., adding new elements to the data, seemed to be a hard operation for novice subjects. The traditional spreadsheet paradigm utilizes a fixed grid of cells in its user interface. Adding new cells to the grid moves old areas to right or down and the new area looks good. But all the other input data or calculation areas that were stretching over the same rows and columns as the new added area, are mutilated.

In the experiment, this kind of addition of new cells to a particular input structure was done by most of the Excel subjects in such a way that the result looked like new empty cells were added in the middle of the other input structures also. This seemed to confuse the novice subjects, because they did not probably comprehend what did happen. Also, if the added cells were at the end of an old area the formulae was not updated to contain the new cells automatically. The novice subjects often forgot to update the referenced areas. Error types E3, E4, E7 and E9 could be explained by this type of action.

Subjects can comprehend homogeneous areas as objects. The error type E7 (adding new items to the input structures) changed the references in formulae, the formulae differed perceptually and the subjects did not perceive the formula area as a homogenous area although the cells were laid out uniformly. This probably caused them to switch their strategy from copying formulae (used in homogenous areas) to writing formulae one by one. It seems that in Task 3 subjects selected this strategy to correct and add new multiplication. This caused a lot of errors in formula referencing.

Overall the Excel subjects in this study seemed to interpret some type of objects consisting of multiple cells and sometimes they seemed to think that these objects could be used in calculations. This type of interpretation has been verified in empirical studies of spreadsheets earlier (Saariluoma and Sajaniemi, 1989, Sajaniemi et al., 1999). Other empirical work on spreadsheet calculation describes also similar cases, where user feels that the spreadsheet system should calculate using task specific larger structures than one cell (e.g., see subject Sue in Hendry and Green study (Hendry and Green, 1994)).

The Basset error type B1 (missing goals) could be explained by the amount of details in goal descriptions. The distinction between available goals can be small and that could be why a subject could not select the appropriate goal. The higher conceptual level of the structured spreadsheet calculation paradigm could also explain B1. The paradigm forces a user to think globally of the solution, i.e., to plan and design the solution before starting the implementation. This causes a lot of mental load for novices and is against the novices' preferred strategy to think locally of the solution (e.g., see Spohrer et al. (1985) for a discussion of Pascal novice strategies). This error manifests itself in the tasks where there is a chain of goals, i.e., tasks 1 and 4.

The number of concepts that is required to successfully use Basset is quite high. The concepts of structures, the types of the structures, the modification of structure sizes, ready-made goals and the concepts of calculation have to be mastered. The subjects did perform quite well, but in some tasks they seemed to make decisions based on fragmentary knowledge of the structured spreadsheet calculation and some surface level details of the goals in Basset. Studies of novice Pascal programming have demonstrated similar behavior (Bonar and Soloway 1985).

The error type B2, the selection of a wrong goal could also be explained by the larger amount of details to look for in Basset than in Excel even at the beginning level of spreadsheet calculation. The novice user has to master the goals of Basset in order to know what kind of computations there are in the system. The goal descriptions have to be read carefully in order to distinct the goals from each other. Some of the subjects did not pay attention to this and selected wrong goals for the task. In budget task most of the wrong goals were Pairwise Sums instead of Pairwise Difference. It could be argued that the time to learn Basset was too short for mastering all of the goals in the system.

Tool Evoked Errors. Two Excel error types (E1, E10) and four Basset error type (B1, B2, B3, B4) have possible causes that may be explained by the usability features of the two tools used in the experiment.

Both tools have some strange features at their user interface and operations. For example, Excel shows the area to be copied surrounded by a flashing line, which is not a normal feature of MS Windows standard. This seems to be distracting to novice users. Also Basset for example does not ask the user whether she wants to save the spreadsheet if it is not saved and the user opens a new sheet.

This category explains errors that were most likely caused by the low-level details of usage. There were total of two Excel and six Basset tool type error causes found in the experiment. The first Excel tool type error cause was in E1 (sum includes empty cells at end). Excel's AutoSum-feature automatically includes empty cells between the referenced number area and the cell in which the formula is written. This does not produce an immediate error in computation but is a potential error cause for the future use of the sheet. The second Excel tool type error cause was in E10 (malformed decimal number). This happened because some of the subjects confused the usage of comma and period. They used period as a currency value in task 4 and Excel misinterpreted the content of the cell as text. The text value could not be used in computation.

The first and second Basset tool type error causes were in Basset error types B1 (missing goal) and B2 (wrong goal applied) respectively. Basset is still a prototype implementation of the structured spreadsheet calculation paradigm. There are problems with the menu selections, the order of elements in menus and the division of goals into submenus. These problems cause extra mental load, which could be an explanation for these Basset error types in many cases. Basset usability problems with menus were often mentioned at after the course questionnaires.

The third and fourth Basset tool type error causes were in B3 (wrong arguments for the goal). This was a serious error producing many erroneous spreadsheets in tasks one and four. The arguments of a goal are assigned when the goal is applied in Basset. This operation's usability in Basset is poor. A user has to first select the first argument and then select a structure using several user interface operations. After this the user has to select the next argument and so on. The arguments appear in the order they are originally written into the goal specification and this may not meet the requirements of the user.

The fifth and sixth Basset tool type error causes were in B4 (no new item added into the input structure). The error type B4 happened in task 1B but the same subjects could perform the same operation (add new items) correctly in task 3. One likely explanation is that the subjects performed the operation correctly in task 1B but forgot to save the file.

Results summary. The Excel subjects performed significantly better in task 1 than the Basset subjects. This could be explained by strategic differences of the tools. Basset forces a user to work with a more global approach to the solution. The user has to plan beforehand what kind of input structures to use, whether there is an appropriate goal existing etc. Excel allows a user to work at the local cell level, concentrating upon the details of the current cell entry at a time. The local cell level entry strategy seems to work for simple application at the novice level faster than Basset's approach. Also the task 1 required largest amount of cell entries. Entering the cells is slower with Basset than with Excel because Basset requires creating the input structures first and adding the titles require much extra work with multiple menus and dialog windows. This could have slowed Basset users down resulting the numerous incomplete applications.

The Basset subjects performed significantly better in tasks 2 and 3 than the Excel subjects. Accumulation seemed to be a hard calculation for novice spreadsheet users. The Excel subjects used the largest assortment of different plans in task 2. They also made the most errors in this task. The accumulation is a system-implemented goal in Basset. For the Basset subjects it was enough that they understood the goal and could apply it within the system. This explains the difference in groups' performance in task 2. In task 3, inserting new items to existing data and calculation areas in Excel was difficult for the novice subjects because the insertion mutilated other data structures and the cell references in existing formulae. In Basset insertion is an operation applied to an input data structure and the necessary changes in computations will be done automatically by the system. It seems that Basset's higher conceptual level helps the novice users in more complex spreadsheet tasks.

The error classification revealed the largest number of error causes in the class paradigm evoked errors. There were 8 paradigm evoked error types in Excel and 2 in Basset. The overall error numbers were smaller in Basset than in Excel and the classification revealed less paradigm evoked errors than other errors. This result is interesting because it suggests that raising the conceptual level of spreadsheet system benefits the users.

In the doctrine evoked error class, there were 3 Excel error types and 2 Basset error types. These are not so interesting because their reasons cannot be remedied by changing the tools and they disappear when users' expertise in spreadsheet-based computation increases. In the tool evoked error class, there were.2 Excel and 4 Basset error types . This shows that Excel is more mature tool than Basset. These errors are fortunately easy to fix. They just require usability testing and software engineering.

Conclusions

The results show that the error behavior of novice spreadsheet programmers is systematically different between the traditional spreadsheet calculation paradigm and the structured spreadsheet calculation paradigm. The reasons for these error behaviors were considered in the context of the content of subject's mental model of the spreadsheet-based computation doctrine in general, the characteristic differences between the two spreadsheet calculation paradigms, and the usability of the two tools used. We classified the error types into these classes and discussed the probable causes for each error type.

There were 8 paradigm evoked error types in Excel and only 2 in Basset. To our opinion this shows that Basset has succeeded in raising the conceptual level of spreadsheet calculation preventing some of the problems of the traditional spreadsheet paradigm to occur in structured spreadsheets. The remarkably low error rate in Basset users group for paradigm evoked errors makes it evident that chancing the conceptual level of spreadsheet systems will have a significant effect on the number and the type of errors novice spreadsheet users will produce.

In doctrine evoked error class, there were 3 Excel error types and 2 Basset error types. These are errors that can only be remedied by learning and experience. This was a study of novice subjects and the difference between the paradigms in this error class is not great. There were only 2 Excel and 4 Basset error types in the class of tool evoked errors. These are the easiest error causes to fix because they require only minor changes in the user interface of the tools.

The traditional spreadsheet calculation paradigm has been criticized for it's low conceptual level. It has been argued that this low conceptual level causes many errors in spreadsheet development and use. Our study supports this view by finding a large number of traditional spreadsheet paradigm evoked errors. Yet there has not been many attempts to change the conceptual level of spreadsheet calculation (Napier, Lane, Batsell, & Guadagno 1989). The structured spreadsheet calculation paradigm is one effort to help the users of spreadsheet calculation to avoid errors by raising the conceptual level of spreadsheets.

The current study clarified some error types in both the traditional and the structured spreadsheet paradigms. This provides a promising area for further investigation of the underlying reasons for these error types, the advancement of both paradigms and the development of better tools to support spreadsheet users.

Acknowledgements

The author would like to thank Jorma Sajaniemi for his deeply thought advice and warm encouragement during this work, Kari Hassinen, Marja Kuittinen and Jorma Sajaniemi for their insightful explanations of the spreadsheet errors, and Juha Hakkarainen for running one half of the experiment sessions.

References

Bonar, J. G. (1985). Understanding the Bugs of Novice Programmers. Ph. D. Thesis, *University of Massachusetts*.

Bonar, J. G., Soloway, E. (1985). Preprogramming Knowledge: A Major Source of Misconceptions in Novice Programming. *Human-Computer Interaction,* 1(2), 133-161.

Brown, P. S., & Gould, J. D. (1987). An Experimental Study of People Creating Spreadsheets. *ACM Transactions on Office Information Systems,* 5(3), 258-272.

Doyle, J. R. (1990). Naive users and the Lotus interface: a field study. *Behaviour & Information Technology,* 9(1), 81-89.

Hassinen, K., Sajaniemi, J., & Väisänen, J. (1988). Structured Spreadsheet Calculation, *Paper presented at the 1988 IEEE Workshop on Languages for Automation,* 129-133.

Hendry, D. G., & Green, T. R. G. (1994). Creating, Comprehending and Explaining Spreadsheets: a Cognitive Interpretation of What Discretionary Users Think of Spreadsheet Model. *International Journal of Human-Computer Studies,* 40, 1033-1065.

Kay, A. (1984). Computer Software. *Scientific American*, 251(3), 41-47.

Lewis, C., & Olson, G. M. (1987). Can Principles of Cognition Lower the Barriers to Programming, *Paper presented at the Empirical Studies of Programmers: Second Workshop,* Washington, D. C.

Microsoft. (1994). Microsoft Excel User's Guide ver. 5.0. Ireland: Microsoft Corporation.

Napier, H. A., Lane, D. M., Batsell, R. R., & Guadagno, N. S. (1989). Impact of a Restricted Natural Language Interface on Ease of Learning and Productivity. *Communications of the ACM,* 32(10), 1190-1198.

Norman, D. A. (1986). Cognitive Engineering. In D. A. Norman & S. W. Draper (Eds.), User-Centered System Design (pp. 31-61). Hillsale, NJ: Lawrence Erlbaum.

Panko, R. (1997). Spreadsheet Research (SSR) http://panko.cba.hawaii.edu /ssr/. (5.1.2000).

Ronen, B., Palley, M. A., & Lucas, H. C. J. (1989). Spreadsheet Analysis and Design. *Communications of the ACM,* 32(1), 84-93.

Saariluoma, P., & Sajaniemi, J. (1989). Visual information chunking in spreadsheet calculation. *International Journal of Man-Machine Studies,* 30, 475-488.

Sajaniemi, J. (2000). Modeling Spreadsheet Audit: A Rigorous Approach to Automatic Visualization. *Journal of Visual Languages and Computing* 11(1), 49-82.

Sajaniemi, J., Tukiainen, M., & Väisänen, J. (1999). Goals and Plans in Spreadsheet Calculation (Technical Report A-1999-1). Joensuu: *Department of Computer Science, University of Joensuu.*

Spohrer, J. C., Soloway, E., & Pope, E. (1985). A Goal/Plan Analysis of Buggy Pascal Programs. *Human-Computer Interaction,* 1, 163-207.

Tukiainen, M. (1996). ASSET: A Structured Spreadsheet Calculation System. *Machine-Mediated Learning,* 5(2), 63-76.

Appendix A:

Legend: D = Doctrine, P = Paradigm, T = Tool, e = Excel, b = Basset.

De1 Subject does not comprehend the difference of a reference to a cell and a constant with the same value. It is easier to write a constant number 12 than a reference =A7.

De2 Subject comprehends spreadsheets to consist of constants and calculations (formulas) where the distinctive feature of calculations is that they include operators or function calls. The formula =A7 does not have operators, so it may be perceived as not being calculation and so the subject uses the constant number 12 instead of the formula.

De3 Subject has not enough knowledge to implement the right formula. Especially in task 2 the cumulative sum is a cognitively quite hard plan. We saw a total of seven different ways to produce the right result and six incorrect versions of these ideas.

De4 Subject has no mathematical model for the computation needed, so she cannot implement it with a spreadsheet

De5 Subject does not have the right mathematical model to calculate the currency exchange, so she cannot implement the computation.

Pe1 No affect in computation. Subject perceives empty cells as white space and includes them into the calculation, because there is nothing in the cells that could affect the sum calculation. (This does not result an error in computation, but can be a potential error cause in future use of the application)

Pe2 Wrong symmetry demand. The referenced area for the sum of the first input sequence is enlarged to the same size as in the case of the second input sequence (i.e., four cells), although the size of the first input sequence is only three cells.

Pe3 Misunderstanding of Copy-operation. The sum formula of the second input sequence (four cells), is copied as the sum of the first input sequence, although the size of the first input sequence is only three cells.

Pe4 Misusage of Copy-operation. The formula for the sum of the first (smaller) input sequence is copied for the second sequence, but the referenced area is not enlarged. The subject perceives the input sequences as objects that the computation should use, so copying a formula referring to one structure should work for another structure.

Pe5 Adding a new item to the first input sequence using Insert rows-operation accidentally adds a new item to the middle of the second input sequence because the sequences were laid out horizontally next to each other. This new empty cell is automatically added to the referenced area of the sum of the second input sequence. Subject comprehends adjacent cell areas as independent objects and thinks that a spreadsheet system operation affects the individual object only.

Pe6 Subject does not know any other way to insert new items to an input structure than using Insert rows-operation. Possible work around is to lay out the structures vertically, as was done in almost all of the error-free application.

Pe7 Subject does not care of extra empty cells in the reference area of a function as they do not affect the result.

Pe8 Some insertion operations are automatically reflected in the computation (i.e., if the insertion is at the middle of the area), so a subject may think that all insertions are reflected in the computations, although adding a new item to the end of an input sequence requires changing the formulae referencing this area

Pe9 Subject perceives the data areas as whole objects that the computation uses, so adding a new item to a object should be reflected in the computations accordingly.

Pe10 Misusage of the Copy-operation. Subject might think that she has made a dynamic copy of the constant value cell using Copy-operation (so that if the cell content changes, the copy will change also).

Pe11 The subject does not use mouse pointing when inserting references to a formula and makes a visual slip when determining the cell reference for typing

Pe12 (Task 3) There were 3 items in both input sequences and a computation for their pair-wise multiplication. The places for new items occurred in the middle of the input sequences. The Insert rows-operation was applied to different places in the input structures, because the structures were laid out horizontally next to each other but not starting from the same row. This caused the cell references in some of the following formulae to change so that they did not refer to successive rows anymore. Thus the successive formulae in output structures did not have a similar visual form anymore. This may have confused subjects to select a wrong formula to copy.

Pe13 The subject gets confused when the references in formulae updated are automatically after inserting new data items into input structures. The references in adjacent cells cross each other. This can confuse the novice subject to believe that the formulae are somehow wrong. The subject chooses a strategy to write over the old formulae by hand so that they will be "right" again. This takes longer time and they have not enough time to complete all formulae.

Pe14 Subject does not understand the concept of absolute references. The calculation should refer to the multiplier cell with an absolute reference and the subject cannot enter it.

Pe15 (Task 3) Subject comprehends homogeneous areas as whole objects. In almost all cases the input areas were constructed correctly. Either this took all the time or adding new items to the input structures changed the references in formulae and this caused subjects extra mental load and prevented them from devising the new formulae.

Te1 Subject separates the data and the calculations with empty cells and then uses Excel's AutoSum-feature, which automatically includes empty cells below the number area.

Te2 Subject uses period instead of comma as decimal number separator in currency values and Excel misinterpreted the content of the cell as text.

Db1 Subject does not succeed in constructing the right goal for the task.

Db2 Subject's mental model for mathematical operation does not differentiate which of the arguments is the subtrahend and which one is the minuend.

Pb1 Distinction between goals is too delicate. Appropriate goal is not identified among related goals.

Pb2 The distinction between available goals is so small that a wrong goal is selected.

Tb1 Subject looks for the goal in the wrong submenu. In Task 1 the Pairwise Difference goal cannot be found in the Table submenu

Tb2 The goals are listed in a single long list without any order (e.g., alphabetic) and the goal Pairwise Difference is at the end of the list of over 10 goals, so the subject can forget what goal she is trying to select.

Tb3 The argument selection is a tricky operation. It is easy to make the selection in an erroneous way.

Tb4 The wording used in the implementation of computation selection is not natural for the subject. People do not usually "select arguments for subtraction operation", they say "a-b".

Tb5 Subject does not know how to add a new item into the existing structure.

Tb6 Subject does not save the file after the change. Basset does not remind users to save the changes when quitting or opening a new sheet.