Information & Management 45 (2008) 312–320

Contents lists available at ScienceDirect

Information & Management

journal homepage: www.elsevier.com/locate/im

# An auditing protocol for spreadsheet models☆

Stephen G. Powell *, Kenneth R. Baker, Barry Lawson

*Tuck School of Business, Dartmouth College, Hanover, NH 03755, USA*

ARTICLE INFO

*Article history:*
Received 8 January 2007
Received in revised form 1 December 2007
Accepted 28 March 2008
Available online 3 June 2008

*Keywords:*
Spreadsheets
Spreadsheet errors
End-user computing
Auditing
Auditing software

ABSTRACT

Errors are prevalent in spreadsheets and can be extremely difficult to find. A number of audits of existing spreadsheets have been reported, but few details have been given about how the audits were performed. We developed and tested a new spreadsheet auditing protocol designed to find errors in operational spreadsheets. Our work showed which auditing procedures, used in what sequence and combination, were most effective across a wide range of spreadsheets. It also provided useful information on the size and complexity of operational spreadsheets, as well as the frequency with which certain types of errors occur.



## 1. Introduction

Errors are a major problem in traditional software programming, and methods have been developed to find and correct them [23]. Analogous methods are rarely used for spreadsheet models, possibly because the typical spreadsheet developer is unaware of the number and significance of errors or unaware of effective testing procedures. In fact, according to our recent survey [22], fewer than 10% of Excel experts used auditing software. An earlier study [4] showed that spreadsheet users generally did not use such features as built-in auditing tools.

Practitioners have recommended many different approaches to testing a spreadsheet for errors: using extreme inputs, reviewing each formula, sensitivity testing, etc. Some stress the use of *tools*, while others stress the use of *people*. Although some audit results have been published, few details have been given as to how the audit was conducted. After 25 years of spreadsheet use by millions of people, we cannot say which auditing methods work best for which types of spreadsheets and developers.

As part of a research effort on how users and their organizations work with spreadsheets, we developed an explicit auditing protocol and tested it on a significant number of operational spreadsheets. Here we describe the protocol and show how it improves auditing procedures.

## 2. Previous work on spreadsheet audits

Two main approaches characterize research on spreadsheet auditing: *field audits* and *laboratory experiments*. The first involves testing spreadsheets that are already being used in organizations: the auditor does not know how many errors exist in a spreadsheet application or where they are located. Sometimes the auditor has access to the spreadsheet developer and can ask for clarification about its purpose and design as well as the accuracy of its details. Otherwise, the auditor may have to test the spreadsheet without access to the developer. Some, but not all, field audits use auditing software.

In a laboratory experiment the researcher creates spreadsheets and seeds them with errors. The task of the subjects in the experiment is to locate these errors. In some cases the subjects are given instructions for conducting their audits; otherwise they are left to their own devices. Laboratory experiments can also be used to evaluate and improve auditing software.

Much of the literature on spreadsheet errors and auditing [20,21] is concerned with the errors that are found rather than the procedures that are used. Although many authors have offered advice on auditing a spreadsheet, no studies have compared alternative auditing approaches on operational spreadsheets.

### 2.1. Field audits

In an analysis of the incidence of errors, Panko [16] cited seven reports on field audits of spreadsheets. The earliest was Davies and Ikin [7], who tested 19 operational spreadsheets from 10 different organizations but provided no details on how the audits were

☆ This work was performed under the sponsorship of the U.S. Department of Commerce, National Institute of Standards and Technology. Reproduction of this article, with the customary credit to the source, is permitted.

* Corresponding author. Tel.: +1 603 646 2844; fax: +1 603 646 1308.
*E-mail address:* sgp@dartmouth.edu (S.G. Powell).

0378-7206/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.im.2008.03.004

conducted. Cragg and King [6] inspected 20 operational spreadsheets from 10 companies. One person spent an average of 2 h in testing a spreadsheet, but the authors gave no details about their testing methods. Panko [18] reported on the audit of a large-scale capital budgeting spreadsheet at NYNEX. Each of its six main modules was audited by a three-person team, which had access to the developer. The team verified formulae and checked cell references. One cell in each column was studied in detail, and the others in the row were checked for consistency. Test data were used to audit some parts of the module, and Excel's formula auditing tool was used. Clermont [5] used specially developed auditing software in a field audit of three large spreadsheets used by the accounting department of an international firm. The auditor first discussed each workbook, then spot-checked the spreadsheet for errors. Finally, the software tool was run, and highlighted cells were investigated. All irregularities were shown to the developer and classified as errors if the developer agreed.

The most detailed description of an auditing procedure yet published comes from HM Customs and Excise, the tax agency in the United Kingdom [1,2]. This procedure used a software tool (SpACE) created for government auditing of small-business tax returns. The process has been documented by HM Customs and Excise [11]. Because the tax auditors are faced with auditing thousands of spreadsheets, their first goal was to select a subset of spreadsheets to audit. Accordingly, the auditor could terminate an audit under three conditions: if the likelihood of significant errors was judged to be low, if their impact was judged to be low, or if the resources required for a full audit were excessive. Under this procedure, detailed inspection of a spreadsheet was performed on a small subset of candidates. When a specific spreadsheet was selected for auditing, the procedure worked as follows: first, the auditor identified the chain of cells from inputs to output and used the software to follow the chain of dependent cells so that the key formulae could be checked. Then the auditor checked copied formulae for correctness. Finally, 14 types of high-risk cells were checked for arithmetic and logic.

This procedure was designed for the restricted domain of auditing small-business tax returns. Thus all the spreadsheets tested related to the same area of application. In addition, it assumed that only a subset of incoming spreadsheets could be tested in detail, so the procedure focused on identifying high-risk candidates. The goal of the Customs and Excise procedure was quite different from ours, which was to develop a *general-purpose* auditing procedure that could be applied effectively to a spreadsheet of any size, complexity, and origin. Therefore, our study differed in several ways from theirs and the others. First, we used commercially available software; second, we examined a large number of spreadsheets across many organizations; and finally, we did not have access to the developers.

### 2.2. Laboratory audits

Laboratory experiments typically employ spreadsheets into which a small number of errors have been planted. Galletta et al. [9] devised an experiment with six simple spreadsheets and concluded that subjects with accounting expertise found more errors than others and that subjects with spreadsheet expertise found errors faster. Galletta et al. [10] studied the effect of presentation style and found that subjects who had access to the spreadsheet formulae did not perform better than those who saw only the resulting numbers. Panko and Sprague [19] examined the capability of students to find their own errors. The study suggested that the native ability of developers to correct their own errors may be limited. Panko [17] studied error-finding by auditors working individually and in groups and found that groups tended to simply pool the errors already found by their individual members. Teo and Lee-Partridge [24] studied the error-finding abilities of student subjects in spreadsheets with both quantitative and qualitative errors. Their experiments indicated that mechanical errors were most easily detected, followed by logic and omission errors. Qualitative errors, however, proved much more difficult to detect. Howe and Simkin [12] investigated some demographic factors that might help explain error-detection ability, but the only general conclusion that could be drawn from their analysis was that formula errors were significantly more difficult to detect than other types. Janvrin and Morrison [13] found that a structured design approach reduced errors in an experimental setting.

Auditing software has also been tested against spreadsheets with seeded errors. Davis [8] conducted experiments with students to determine whether two tools (a flowchart-like diagram and a data dependency diagram) were useful. His results showed that both tools were judged to be better than nothing in investigating cell dependencies, and that the data dependency tool was better than the built-in Excel tools. Nixon and O'Hara [15] compared the performance of five auditing tools in finding seeded errors. The most successful identified over 80% of them. The mechanisms that were most helpful provided a visual understanding of the schema or overall pattern of the spreadsheet, and those that searched for potential error cells. A major limitation of this study was that the tools were tested by the researcher, who knew the location of the errors. Chan et al. [3] built four software tools for visualizing precedent/dependent relationships in a spreadsheet. They did not test these tools experimentally but suggested different ways in which they could be used. Kruck [14] developed three aids for building accurate spreadsheets.

How transferable are these results from the laboratory to the real world? Operational spreadsheets are usually larger and more complex, and built by subject-area experts to be used over an extended period of time. Also, errors in operational spreadsheets are not known to auditors and the user environment is different from a laboratory. Spreadsheet developers are likely to be more experienced both in the use of spreadsheets and the area of expertise, more motivated, more closely monitored, etc. Spreadsheets used in organizations may improve over time as errors are found.

Our review of the literature [20] identified several shortcomings in both field audits and laboratory studies. Few studies have reported *how* the audits were carried out, they have not tested different approaches to compare their effectiveness, and they have not reported details of the spreadsheet sample, such as size, complexity, or application area. Laboratory studies are confined to simple spreadsheets and the results may not be transferable to practice in the field. A number of important questions therefore remain:

- Are spreadsheet auditing tools effective?
- Are particular functions or types of formulae prone to errors?
- What sequence of steps is most effective in identifying errors?
- How common are errors?
- Are particular auditing tools especially effective in identifying certain types of errors?

## 3. Research design

Our auditing protocol was designed to satisfy five criteria:

(1) intended for completed, operational spreadsheets;
(2) usable by any moderately experienced users;
(3) applicable to spreadsheets of any size and complexity;
(4) suited to spreadsheets from any area of application;
(5) usable without access to the spreadsheet developer.

### 3.1. Spreadsheet sample

Our auditing protocol was tested on more than 100 operational spreadsheets during its development. In the initial stages, we audited a small number of spreadsheets and debriefed the auditors to learn what parts worked and what did not. We thus revised the protocol many times.

Some of our test spreadsheets were drawn from organizations, such as consulting companies, a bank, a college, a state government agency, and a large energy firm; others were gathered via the web. In all cases, they were completed spreadsheets that had been in use for some time. While our sample was not *random*, it contained a wide variety from a general population. The sample included spreadsheets created by both novice and expert developers and that spanned the range from small and simple to large and complex.

### 3.2. Auditor training

Our spreadsheet auditors were current undergraduate or graduate students in business or engineering or recent alumni of these programs. All had had several years experience with Excel, usually in a business setting. None was a professional programmer or spreadsheet developer.

Novice auditors first studied the protocol, which described the stages of an audit and the data to be gathered. Then each was given two or three spreadsheets to audit. Their workbooks were then reviewed by the authors for adherence to the auditing protocol and for quality of the audit. On average, auditor training took 10 h.

### 3.3. Auditing software

Our protocol used two software tools: *XL Analyst* (http://www.xlanalyst.co.uk/) and *Spreadsheet Professional* (http://www.spreadsheetinnovations.com/). These were selected from a list of about 50 tools compiled by Roger Grinde of the University of New Hampshire.

*XL Analyst* is an Excel add-in that evaluates 28 aspects of a spreadsheet, from "Formulas evaluating to an error" to "Use of SUMIF." A description of the *XL Analyst* report is given in Table 1. We selected this analyzer for its simplicity: it runs a single-pass analysis of a workbook and creates a summary. It also offers numerical estimates of the workbook size and complexity. Finally, it provides an Overall Risk Rating based on a weighted average of the measured factors. One limitation of this tool was that it provided only a flag when a risk condition was met and the address of the single cell involved, but it did not report *how many* cells met it or their locations.

*Spreadsheet Professional* is a collection of tools for building, testing, analyzing, and using spreadsheets. In our auditing protocol we made use of two of its features: *maps* and *calculation tests*. The mapping tool created a coded version of each worksheet. Each non-blank cell was coded as a label, a number, or a formula. It also showed which formulae had been copied from an original formula. A sample map is shown in Fig. 1. The calculation test tool checked the workbook for the 25 conditions given in Table 2. For each of these categories it reported the number of cells involved and their addresses.

We selected it from among several competing products, all of which appeared to be mature and offered a rich suite of tools. An advantage to us was that *Spreadsheet Professional* was an add-in to Excel rather than stand-alone.

### 3.4. Auditing protocol

The auditing protocol involved the following eleven steps (the complete protocol is available at http://mba.tuck.dartmouth.edu/spreadsheet/index.html):

**Table 1**
*XL Analyst* report

| |
|---|
| Factors suggesting a high risk of an error |
| Circular references |
| Cells displaying a number but storing text |
| Mixed formulae and values |
| Formulae evaluating to an error |
| VLOOKUPS expecting an ordered list |
| HLOOKUPS expecting an ordered list |
| Factors suggesting a significant risk of an error |
| Links to external workbooks |
| Presence of very hidden sheets |
| Hidden rows or columns |
| "=+" construct |
| Conditional formatting |
| Use of pivot tables |
| Factors suggesting complex logical modelling |
| Array formulae |
| Nested IF statement |
| Use of SUMIF |
| Use of database functions (DSUM, etc.) |
| Use of INDIRECT |
| Measures |
| Longest formula |
| Most complex formula |
| Total number of formulae |
| Total number of unique formulae |
| Workbook size |
| Number of worksheets |
| Total all lines of VBA code |
| Largest formula result |
| System messages |
| Protected worksheets |
| Protected workbook structure |
| Other |

1. Run the two software tools.
2. Transfer selected results from the software tools to a data record sheet.
3. Record the purpose of the workbook and each worksheet.
4. Examine workbook for use of Excel functions.
5. Review the results of *XL Analyst* and use them to locate errors.
6. Review the *Spreadsheet Professional* maps and use them to locate errors.
7. Review the *Spreadsheet Professional* calculation tests and use them to locate errors.
8. Review all formulae not already reviewed for errors.
9. Conduct various sensitivity analyses to uncover errors.
10. Rate the workbook on various aspects of spreadsheet design (e.g., use of modules).
11. Record the total time taken by the audit and record comments on special situations encountered.

This sequence of steps evolved during months of development. We initially focused on the chain of cells used to calculate the output, as suggested by the H.M. Customs and Excise procedure. However, in many cases we encountered hundreds of outputs but could identify no single logical chain. We therefore developed a *layered* approach, in which we used the auditing tools to gain an understanding of the physical and logical layout of the spreadsheet and to identify high-risk cells, and only later did we examine individual formulae.

During our protocol design we trained auditors and tested the protocol ourselves on dozens of operational spreadsheets. Many changes were made to the protocol over this time period. We had initially thought that performance testing (finding the effect of different inputs on outputs) would be effective in locating errors

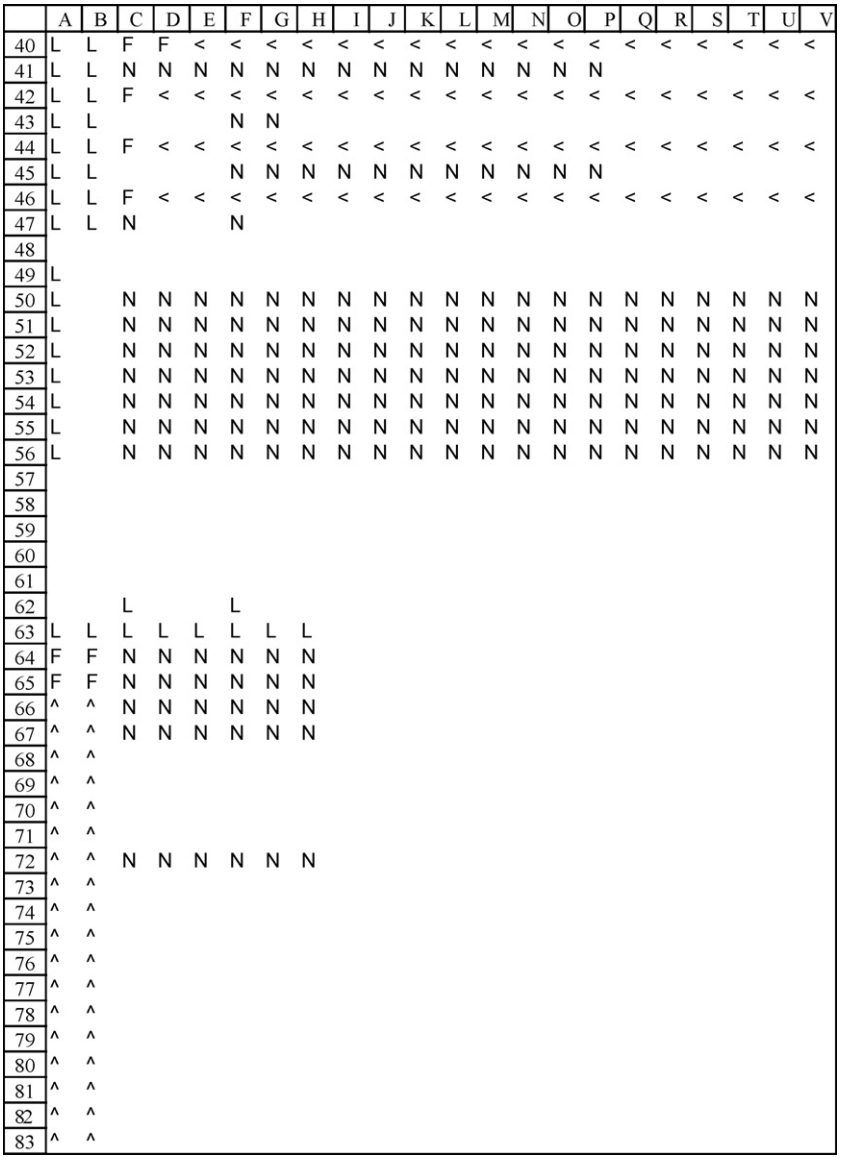

**Fig. 1.** Sample map report from *Spreadsheet Professional*.

and that this should be conducted early in an audit. We based this on our own experience on spreadsheets primarily developed for decision making. However, we found that performance testing was often either ineffective or impossible; in many operational spreadsheets it is difficult to distinguish inputs and outputs from other numbers. Ultimately, we found it to be relatively ineffective in locating errors in our sample.

We also found that certain options in the use of the auditing software could make a significant difference in our results. For example, *Spreadsheet Professional* allowed the user to choose one of three alternative definitions of a unique (i.e., uncopied) formula. One alternative recognized copying across rows, a second recognized copying down columns, and the third recognized both types of copying. This choice had a major influence on the number of cells flagged as potential errors, although not on the number of errors found. Some auditors preferred to work with large numbers of potentially problematic cells highlighted by these tests; others preferred combing through fewer false positives. In the end we decided which would be most effective across all spreadsheets and auditors and incorporated that choice in the final protocol.

### 3.5. Data collection

All information collected during our audits was recorded in a standard format on a single worksheet. After recording the name and purpose of the workbook, the auditor recorded the name of each worksheet along with all those sheets that were linked to it.

Numerical information produced by *XL Analyst* was then captured. This included:

- Overall risk rating.
- Longest formula (characters).
- Most complex formula (operations).
- Total number of formulae.
- Total number of unique formulae.
- Percent unique formulae.
- Workbook size.
- Number of worksheets.
- VBA Code (line/components).
- Largest formula result.

**Table 2**
Calculation tests in *Spreadsheet Professional*

| |
|---|
| 1. Unused input values |
| 2. Unused calculations |
| 3. No precedents |
| 4. Dependents rule |
| 5. Blank cells references |
| 6. Errors referenced |
| 7. Non-numeric cell referenced |
| 8. Forward row reference |
| 9. Forward column reference |
| 10. Hidden cell referenced |
| 11. Range name |
| 12. Duplicate range names |
| 13. External references |
| 14. IF function |
| 15. Double IF function |
| 16. NPV function |
| 17. VLOOKUP function |
| 18. HLOOKUP function |
| 19. LOOKUP function |
| 20. Numeric rule: numbers in formula |
| 21. Complex calculation |
| 22. Unprotected calculation |
| 23. Lotus evaluation rules |
| 24. Worksheet protection |
| 25. Calculation manual |

*Spreadsheet Professional* also generated numerical data, but at the individual worksheet level. This data was captured at the worksheet level and then summarized for the workbook as a whole. This included:

- Number of numeric inputs.
- Number of formulae.
- Number of unique formulae.
- Percentage of unique formulae.
- Number of labels.

Next the auditor examined each worksheet for the use of built-in functions and recorded the names of the functions by type. The names were then concatenated in a single cell for further analysis.

Both *XL Analyst* and *Spreadsheet Professional* checked individual cells for a number of conditions that may have indicated errors or problems. *XL Analyst* checked for the 17 conditions shown in Table 1. *XL Analyst* reported only the cell address of a single cell that met a given condition, even if dozens of other cells did so also. *Spreadsheet Professional* also checked for potential errors using the 25 calculation tests shown in Table 2. It reported all the cells in each worksheet that satisfied any of the tests. We recorded the cell addresses of flagged cells and calculated the total number of cells identified.

The auditors next recorded their subjective valuation of the design qualities of the workbook by rating the following eight qualities on a 1–5 Likert scale:

- Overall ease of understanding.
- Use of modules.
- Use of parameterization.
- Use of range names for parameters.
- Use of range names for formulae.
- Ease of use.
- Ease of communication.
- Overall technical quality.

The auditors then recorded how the workbook was documented, looking for evidence of six methods:

- Model assumptions—explanation of major assumptions behind the model.
- Sources for inputs—sources given for numerical inputs.
- Guide to sheets—overview of the purpose of sheets in workbook.
- Cell comments—comments in individual cells.
- Pseudocode for formulae with explanation for them, such as "IF (Demand > Supply, Supply, Demand)".
- Notes in cells—text in cells explaining formulae or assumptions.

Next the auditors looked for evidence that the following security tools were used:

- Protected cells.
- Hidden cells.
- Data validation.

We then collected data on errors, defining six categories of errors:

- Logic—formula uses incorrect logic.
- Reference—formula refers to wrong cell(s).
- Hard-coding—number(s) appear in formula.
- Copy/Paste—formula error due to misuse of copy/paste.
- Data input—wrong input data.
- Omission—factor omitted from formula.

We recorded errors by *instance*, not by *cell*. An instance of an error represents a single conceptual error, which may be repeated in other cells. For example, if a SUM formula in cell D46 points to the wrong input range, we would classify it as a *Reference* error. If that same erroneous formula appeared in cells E46:G46, we would count that as one error instance with four error cells.

For each error instance we recorded the following information:

- cell address(es);
- number of cells;
- whether identified by numerical tests in *XL Analyst* or *Spreadsheet Professional*;
- type of error;
- how it was discovered;
- explanatory comments.

## 4. Quantitative results

Our primary motivation for developing a systematic approach to auditing was to identify errors. As a side benefit, however, our audits produced quantitative measures of size, complexity, function use, and other aspects of our sample spreadsheets. This information was useful because it allowed us to summarize our sample with measures that could be compared to other samples of spreadsheets and allowed us to estimate revealing aspects of operational spreadsheets such as the percentage of unique formulae.

### 4.1. Spreadsheet size and complexity

Table 3 summarizes data on our sample of 50 spreadsheets in 12 dimensions, as measured by *XL Analyst* and *Spreadsheet Professional*. The median number of kilobytes filled by these models was 189, with a range from 28 to 3852.

Perhaps the best single measure of complexity and degree of effort required to audit a spreadsheet is the number of formulae involved. The median number of formulae in our sample was 1294, with a range from 25 to 63,371. The software also measured the

**Table 3**
Quantitative measures of sample spreadsheets

| Measure | Median | Minimum | Maximum |
|---|---|---|---|
| *XL Analyst* | | | |
| Longest formula (number of characters) | 114 | 15 | 711 |
| Most complex formula (operations) | 14 | 3 | 120 |
| Total number of formulae | 1294 | 25 | 63,731 |
| Total number of unique formulae | 105 | 9 | 1,685 |
| Percent of unique formulae (%) | 10.0 | 0.3 | 56.1 |
| Workbook size (kb) | 189 | 28 | 3,852 |
| Number of worksheets | 5 | 1 | 44 |
| *Spreadsheet Professional* | | | |
| Number of numeric inputs | 562 | 21 | 44,470 |
| Number of formulae | 1294 | 25 | 63,731 |
| Number of unique formulae | 193 | 11 | 4,081 |
| Percentage of unique formulae (%) | 24.7 | 0.6 | 97.9 |
| Number of labels | 417 | 42 | 91,137 |

number of *unique* formulae: this eliminated formulae that were copied from others. *XL Analyst* reported a median of 105 unique formulae, with a range from 9 to 1685. *Spreadsheet Professional* reported a median of 193 unique formulae, with a range from 11 to 4081. (Differences in counts of unique formulae are to be expected, as the algorithms used to detect copying have not been standardized.) The percentage of unique formulae has a median of 10.0% (*XL Analyst*) or 24.7% (*Spreadsheet Professional*), with ranges from 0.3% to 56.1% and 0.6% to 97.9%, respectively.

*XL Analyst* also recorded data on the longest and most complex formulae in a workbook. The median number of characters in the longest formula was 114, with a range from 15 to 711. The median number of operations in the most complex formula was 14, with a range from 3 to 120.

These results suggested several facts about our sample:

- Individual spreadsheets ranged from very small and simple to very large and complex.
- The median spreadsheet size was quite large, whether measured in terms of worksheets, kilobytes, or number of formulae.
- The number of unique formulae was generally a small percentage of the total number of formulae, indicating a high incidence of copied formulae.
- Long and complex formulae occurred in a large percentage of spreadsheets.

### 4.2. Use of functions

Little has been reported about how often functions are used and which ones are most often used. As part of our protocol we required the auditors to record the functions used on each sheet in the workbook. (We did not determine the number of times it was used.)

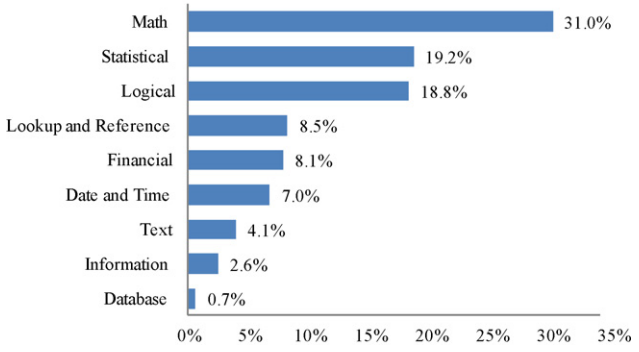


**Fig. 2.** Frequency of function use by type.

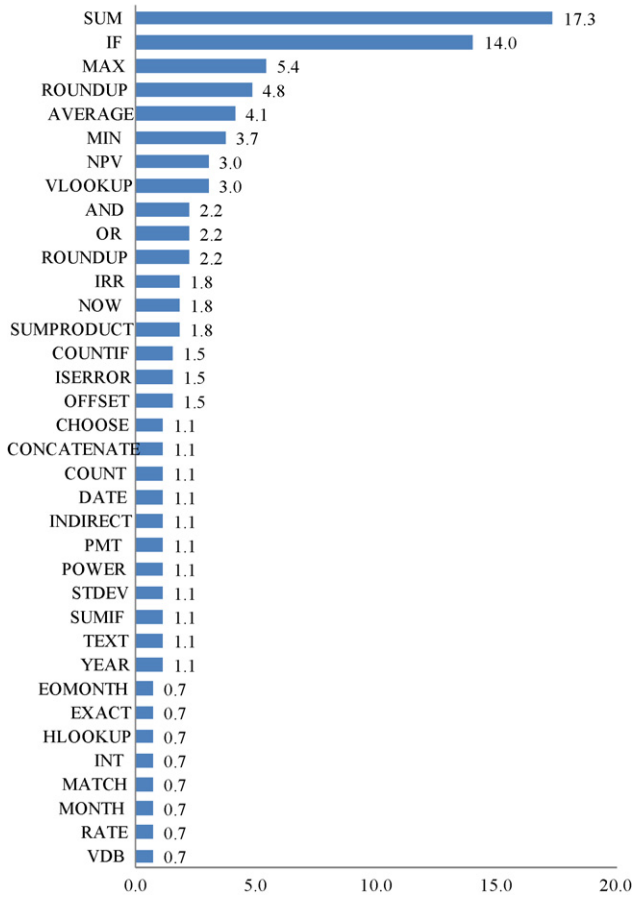


**Fig. 3.** Frequency of function use by individual function.

Fig. 2 shows the results by type of function and Fig. 3 by individual function. In total, we identified 65 distinct functions used in our sample, but only 11 of these appeared in 6 or more worksheets. The most common function was the SUM function, appearing at least once in 17.3% of worksheets.

### 4.3. Errors

In 3 of the 50 spreadsheets audited we did not find any errors of the types in our auditing protocol. In the remaining 47 we found a total of 483 instances of errors involving a total of 4855 error cells. On average, each error instance involved 10.05 cells. The average cell error rate over all 270,722 formulae audited was 1.79%. For the 47 spreadsheets with errors, the minimum number of error instances was 1 and the maximum was 65. The median number was 7, and most spreadsheets had 10 or fewer. The minimum number of error cells was 1 and the maximum was 1711. The median was 41, and most spreadsheets had 100 or fewer. Fig. 4 shows how the error instances and error cells were distributed by error type.

## 5. How errors were discovered

One of our goals was to determine the auditing procedures that were most effective in identifying errors. *XL Analyst* and *Spreadsheet Professional* both flagged potentially problematic cells, and these were systematically investigated by our auditors. In addition, *Spreadsheet Professional* provided a mapping tool that

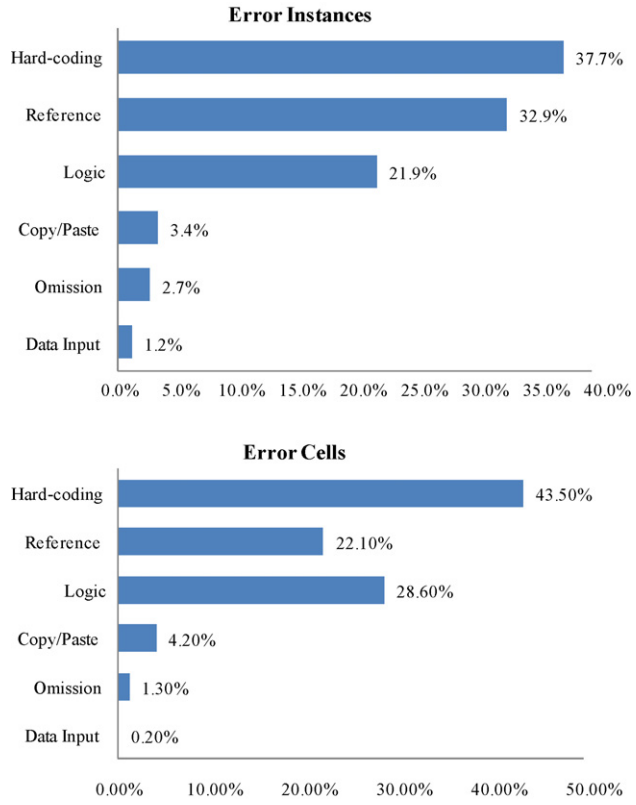


**Fig. 4.** Errors categorized by type.

coded each non-blank cell as a label, a number, or a formula. It also showed which formulae had been copied from an original one. Some errors stand out in these maps, and our auditors were trained to examine them. Finally, the protocol required a careful inspection of all formula cells that had not already been investigated (we refer to this as *code inspection*).

Fig. 5 shows how the error instances and error cells were identified. Some of these results can be attributed to the sequence in which we used the tools. For example, we would have identified more errors using code inspection if we had performed code inspection before auditing. However, as a result of our experience we believed that starting an audit with a *Spreadsheet Professional* map analysis and calculation tests was more effective.

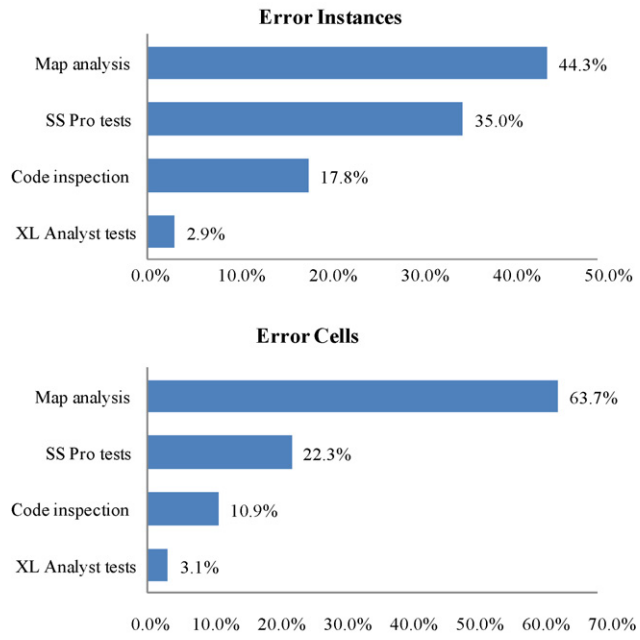


**Fig. 5.** How errors were identified.

Table 4 categorizes error instances and error cells by type of error and how they were discovered. These results suggested several conclusions. First, map analysis was a powerful means for identifying errors. It was quick and rarely provided false positives. It worked by revealing the overall structure of a worksheet and by highlighting cells that broke a pattern. Second, the automated tests helped to identify a large proportion of the remaining errors, but the process produced large numbers of false positives. (Fewer would be generated if some of the 25 tests were shut off.) Third, code inspection was necessary, but it identified only a small portion of errors once map analysis and the error tests had been completed.

## 6. True and false positives

Auditing software flagged a subset of the formula cells in a workbook as potential errors. False positives are flagged cells that are not errors; false negatives are error cells that are not flagged. High rates of either make auditing software less effective. We

**Table 4**
Errors by type and how identified

| | Map analysis | *SS Pro* tests | *XL Analyst* tests | Code inspection | Total |
|---|---|---|---|---|---|
| (A) Error instances | | | | | |
| Hard-coding | 93 (51.1%) | 56 (30.8) | 6 (3.3) | 27 (14.8) | 182 |
| Reference | 40 (25.2) | 83 (52.2) | 8 (5.0) | 28 (21.7) | 159 |
| Logic | 59 (55.7) | 24 (22.6) | 0 (0.0) | 23 (21.7) | 106 |
| Copy/paste | 11 (64.7) | 5 (29.4) | 0 (0.0) | 1 (5.9) | 17 |
| Omission | 8 (61.5) | 0 (0.0) | 0 (0.0) | 5 (38.5) | 13 |
| Data | 3 (50.0) | 1 (16.7) | 0 (0.0) | 2 (33.3) | 6 |
| Total | 213 (44.1) | 166 (34.4) | 14 (2.9) | 88 (18.2) | 483 |
| (B) Error cells | | | | | |
| Hard-coding | 1440 (68.2%) | 382 (18.1%) | 127 (6.0%) | 162 (7.7%) | 2111 |
| Reference | 274 (25.5) | 640 (59.6) | 22 (2.0) | 138 (12.8) | 1074 |
| Logic | 1117 (80.4) | 51 (3.6) | 0 (0.0) | 221 915.9) | 1389 |
| Copy/paste | 197 (95.6) | 8 (3.9) | 0 (0.0) | 1 (0.5) | 206 |
| Omission | 60 (92.3) | 0 (0.0) | 0 (0.0) | 5 (7.7) | 65 |
| Data | 7 (70.0) | 1 (10.0) | 0 (0.0) | 2 (20.0) | 10 |
| Total | 3095 (63.8) | 1082 (22.3) | 149 (3.1) | 529 (10.9) | 4855 |

investigated the false positive and negative rates for the calculation tests in *Spreadsheet Professional*. This software flags a cell when it violates one or more of 25 conditions. Thus one cell can be flagged several times.

Our sample (50 spreadsheets) involved 270,722 formulae, of which we classified 4855 as errors. *Spreadsheet Professional* generated 62,719 cell flags in all, of which 6276 referred to one of these error cells. Thus 56,443 flags out of the total of 62,719, or 90%, were false positives. However, error cells were flagged more than once by this software. The 6276 error cell flags identified only 1757 different errors, thus an individual error cell had been flagged an average of 3.6 times. Since *Spreadsheet Professional* flagged 1757 out of a total of 4855 error cells, missing 3089 error cells, the false negative rate was 64%. As one might expect, the calculation tests were more effective in identifying certain types of errors. One might therefore ask how likely a certain flagged cell was to be a certain type of error. For example, *Spreadsheet Professional* generated 316 flags for the condition "Blank cells referenced." Of these 316 cells, 285 were classified as *Reference* errors. Thus a cell flagged for this condition had a 90.2% chance of being a Reference error. Similarly, a cell flagged for *No precedents* had a 67.7% chance of being a *Logic* error. Finally, a cell flagged for *Numeric rule*, *Unused calculation*, *Forward reference*, *IF function*, or *Hidden cell* had an 80–90% chance of being a *Hard-coding* error.

## 7. Auditing time

Our protocol was designed to reveal only certain types of errors, and there was no guarantee that it would find all the errors in any one spreadsheet. However, we still expected that it would use the auditor's time efficiently and that it would not take so much time as to be impractical.

The average time spent by our auditors on our sample spreadsheets, including the time spent recording and categorizing the errors found, was 3.25 h. The range was from 0.8 to 15.3 h. This seemed to be a reasonable amount of time to devote to auditing a spreadsheet of importance to an organization.

We speculated that the time taken to audit a spreadsheet depended on many factors, including its size and complexity and the domain knowledge and auditing skill of the auditor. In our sample, the strongest correlation between auditing time and the various quantitative measures was with the number of worksheets in the workbook. The median time to audit a single sheet was 25 min. Fig. 6 shows the distribution of the time per sheet for all 50 workbooks. While one workbook required nearly 300 min per sheet, most required less than 50 min.

## 8. Recommendations for practice

Our protocol was not intended for use as a day-to-day auditing process in organizations. However, we learned a great deal about effective auditing and the design needs of spreadsheets.

First, auditing software is valuable. *Spreadsheet Professional* especially was a highly effective tool. Its worksheet maps provided a quick understanding of the logical design of each sheet and often pointed to problem cells. The calculation tests, despite a high rate of false positives and negatives, also pointed to many errors. Learning to use such a tool took time and experimentation, but was effective. Second, our auditing experience reinforced our belief that a logical spreadsheet design was critical to avoiding errors. Many of the spreadsheets we audited were astonishingly complex, and were not designed to reduce complexity or make understanding easy. Complexity was a major source of errors. Third, we gained insight into problems needing attention by a spreadsheet auditor: complex formulae are risky; similarly, functions such as IF, VLOOKUP, and NPV were often misused and should be audited carefully. We even found that simple formulae often referred to blank cells or to the wrong cells. Some of these errors were identified by the auditing software while others can be found by code inspection.

## 9. Summary

We developed a general-purpose auditing procedure for operational spreadsheets and tested it on 50 completed spreadsheets taken from a wide variety of sources. Our auditing protocol used two commercially available software tools to improve the speed and effectiveness of the process.

Using the procedure, we documented the size and complexity of a large sample of spreadsheets. We determined the frequency with which built-in functions were used and found errors in about 1.8% of all formula cells.

The auditing software generated a high percentage of false positives and false negatives. However, we believe that auditing software is far more effective in identifying errors than unassisted code inspection.

While we found that operational spreadsheets were often extremely complex, we also found that an effective audit could be

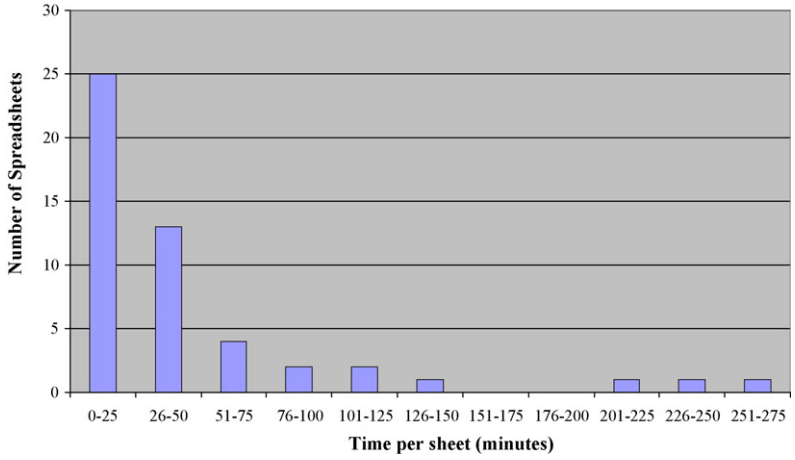


**Fig. 6.** Distribution of auditing time.

conducted in 5–10 h. This could be reduced if the auditor was knowledgeable in the problem domain or had access to the spreadsheet developer. We observed that auditors developed skills that allowed them to understand the formal structure of a complex spreadsheet. They also developed a sense of where errors were likely to occur. Organizations could benefit from training auditing specialists and providing auditing services to spreadsheet developers.

## References


[1] R. Butler, Is this spreadsheet a tax evader? in: Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000, pp. 1–6.
[2] R. Butler, Risk Assessment for Spreadsheet Developments: Choosing Which Models to Audit, H.M. Customs and Excise, UK, 2000.
[3] H.C. Chan, C. Ying, C. Peh, Strategies and visualization tools for enhancing user auditing of spreadsheet models, Information and Software Technology 42 (2000) 1037–1043.
[4] Y.E. Chan, V. Storey, The use of spreadsheets in organizations: determinants and consequences, Information and Management 31 (1996) 119–134.
[5] M. Clermont, A spreadsheet auditing tool evaluated in an industrial context, in: Proceedings of the European Spreadsheet Risks Interest Group Annual Conference, Cardiff, Wales, (2002), pp. 35–46.
[6] P. Cragg, M. King, Spreadsheet modelling abuse: an opportunity for OR? Journal of Operational Research Society 44 (1993) 743–752.
[7] N. Davies, C. Ikin, Auditing spreadsheets, Australian Accountant 57 (11) (1987) 54–56.
[8] J. Davis, Tools for spreadsheet auditing, International Journal of Human-Computer Studies 45 (1996) 429–442.
[9] F. Galletta, D. Abraham, M. El Louadi, W. Leske, Y. Pollalis, J. Sampler, An empirical study of spreadsheet error-finding performance, Accounting, Management & Information Technology 3 (2) (1993) 79–95.
[10] F. Galletta, K. Hartzel, S. Johnson, J. Joseph, S. Rustagi, Spreadsheet presentation and error detection: an experimental study, Journal of Management Information Systems 13 (3) (1997) 45–63.
[11] H.M. Customs and Excise Computer Audit Service, Methodology for the Audit of Spreadsheet Models, 2001.
[12] H. Howe, M. Simkin, Factors affecting the ability to detect spreadsheet errors, Decision Sciences Journal of Innovative Education 4 (1) (2006) 101–122.
[13] D. Janvrin, J. Morrison, Using a structured design approach to reduce risks in end-user spreadsheet development, Information and Management 37 (2000) 1–12.
[14] S. Kruck, Testing spreadsheet accuracy theory, Information and Software Technology 48 (2006) 204–213.
[15] D. Nixon, M. O'Hara, Spreadsheet auditing software, in: Proceedings of the European Spreadsheet Risks Interest Group Annual Conference, Amsterdam, Netherlands, 2001.
[16] R. Panko, What we know about spreadsheet errors, Journal of End-User Computing 10 (1998) 15–21.
[17] R. Panko, Applying code inspection to spreadsheet testing, Journal of Management Information Systems 16 (2) (1999) 159–176.
[18] R. Panko, What We Know About Spreadsheet Errors, http://panko.cba.hawaii.edu/ssr/Mypapers/whatknow.htm (accessed September 2, 2006), 2005.
[19] R. Panko, R. Sprague, Hitting the wall: errors in developing and code inspecting a "simple" spreadsheet model, Decision Support Systems 22 (1998) 337–353.
[20] S. Powell, K. Baker, B. Lawson, A Critique of the Literature on Spreadsheet Errors, Decision Support Systems, in press.
[21] S. Powell, K. Baker, B. Lawson, Errors in Operational Spreadsheets, Journal of Organizational and End User Computing, submitted for publication.
[22] S. Powell, K. Baker, B. Lawson, Spreadsheet Experience and Expertise, Omega, in press.
[23] K. Sommerville, Software Engineering, 7th ed., Addison Wesley, 2004.
[24] T. Teo, J. Lee-Partridge, Effects of error factors and prior incremental practice on spreadsheet error detection: an experimental study, Omega-The International Journal of Management Science 29 (2001) 445–456.


**Stephen Powell** is a Professor at the Tuck School of Business at Dartmouth College. His primary research interest lies in modeling production and services processes, but he has also been active in research in energy economics, marketing, and operations. In 2001 he was awarded the INFORMS Prize for the Teaching of Operations Research/Management Science Practice. He is the co-author with Kenneth Baker of *The Art of Modeling with Spreadsheets* (Wiley, 2004).

**Kenneth Baker** is a faculty member at Dartmouth College. He is currently Nathaniel Leverone Professor of Management at the Tuck School of Business and also adjunct professor at the Thayer School of Engineering. At Dartmouth, he has taught courses relating to decision science, manufacturing management, and environmental management. Over the years, much of his teaching and research has dealt with production planning and control, and he is widely known for his textbook *Elements of Sequencing and Scheduling*, in addition to a variety of technical articles. In 2001 he was named a Fellow of INFORMS's Manufacturing and Service Operations Management (MSOM) Society, and in 2004 a Fellow of INFORMS. He is the co-author with Stephen Powell of *The Art of Modeling with Spreadsheets* (Wiley, 2004).

**Barry Lawson** is a research associate at the Tuck School of Business at Dartmouth and is also a visiting scholar in the geography department of the college. As research associate at Tuck he serves as the program manager for the Tuck Spreadsheet Engineering Research Project.