

■ RICK HESSE, Feature Editor, Pepperdine University

Facing the Problem of Spreadsheet Errors

by Ray Panko, University of Hawaii

On the twenty-fifth anniversary of electronic spreadsheets, it might be well to pause and reflect on the downside of these ubiquitous, handy tools. Ask a programmer if a recently developed program module contains an error. He or she will almost always say, "Certainly!" From the time they are nerdlings, programmers are taught that program modules, *after* careful development, will contain errors in 2% to 5% of all lines of code. They are also taught that extensive (and expensive) testing is necessary to get the error rate down to about 0.1% to 0.4% in shipping software.

Spreadsheet Errors

In contrast, if you ask a spreadsheet developer the same question about a carefully developed spreadsheet with dozens, hundreds, or even thousands of lines of code, he or she will almost always say, "No," and will do so very indignantly. Testing? Forget about it. A long stream of research has shown that very few firms have policies mandating the testing of spreadsheets, and few spreadsheet developers do testing on their own. When they do what they call "testing," furthermore, it usually consists of "looking over the spreadsheet for reasonableness" or some other whiskey cure. (Of all the things that do not cure the common cold, whiskey is the most popular.)

Should spreadsheet developers be so confident? Nearly 30 years of research says no. Lab studies have shown that spreadsheet developers make errors in 2% to 5% of all formulas. (Sound familiar?) This number, furthermore, varies very little between novices and experienced spreadsheet developers.

Most people are more convinced by results from field audits of real-world operational spreadsheets. Table 1 shows some key data from field audits since 1995. The formula error rate in audits that supplied the necessary information was (drum roll) 5.2%.

A formula error rate of 2% to 5% is very small, but spreadsheets tend to be very large. If you have an average formula error rate of e , and if you have N formulas in a chain, then the likelihood of your having an error is $1-(1-e)^N$. Even if you have only a dozen or so formulas, you are quite likely to have an error. If you have hundreds or thousands of unique (non-copied) formulas, the question is *how many* errors you have and *how serious* they are.

Research Data

Table 1 indicates that significant errors are, in fact, common. For example, Coopers and Lybrand found that 91% of financial spreadsheets with more than



Ray Panko

is a professor of IT management in the Shindler College of Business Administration at the University of Hawaii. He has been conducting spreadsheet error research and general human

error research since 1993. The content of this article is based on information available at his spreadsheet research Web site and his human error Web site (see urls below).

panko@hawaii.edu

<http://panko.cba.hawaii.edu/ssr>

<http://panko.cba.hawaii.edu/HumanErr>

ERRATUM

In the previous Classroom feature column (May 2006, v37,n3), the following correction should be made in the first column, last sentence above Figure 1 (thanks to Tom Jones, University of Arkansas, for pointing this out):

Activity C—D is a dummy activity, to indicate that regulations will not allow ~~bags to be loaded until the cabin has been cleaned~~: passengers to be boarded until the bags have been unloaded.

Authors	Year	No. of Spreadsheets Audited	Average Size (Cells)	Percentage of Spreadsheets with Errors	Cell Error Rate	Comment
Hicks	1995	1	3,856	100%	1.2%	One omission error would have caused an error of more than a billion dollars.
Coopers & Lybrand	1997	23	More than 150 rows	91%		Off by at least 5%. At 5%, financial errors are considered to be material.
KPMG	1998	22		91%		Only significant errors that could affect decisions.
Lukasic	1998	2	2,270 & 7,027	100%	2.2%, 2.5%	In Model 2, the investment's value was overstated by 16%. Quite serious.
Butler	2000	7		86%	0.4%	Only errors large enough to require additional tax payments.*
Clermont, Hanin, & Mittermeier	2002	3		100%	1.3%, 6.7%, 0.1%	Computed on the basis of non-empty cells.
Interview I**	2003	~36 / yr		100%		Approximately 5% had extremely serious errors.
Interview II**	2003	~36 / yr		100%		Approximately 5% had extremely serious errors.
Lawrence and Lee	2004	30	2,182 unique formulas	100%	6.9%	30 most financially significant spreadsheets audited by Mercer Finance & Risk Consulting in previous year.
Total/Average		88		94%	5.2%	

Table 1. Spreadsheet auditing results.

*The low cell error rate probably reflects the fact that the methodology did not inspect all formulas in the spreadsheet but focused on higher-risk formulas. However, error has a strong random component, so the practice of not checking all formulas is likely to miss many errors.

**In 2003, the author spoke independently with experienced spreadsheet auditors in two different companies in the United Kingdom, where certain spreadsheets must be audited by law. Each company audited about three dozen spreadsheets per year. Both said that they had never seen a major spreadsheet that was free of errors. Both also indicated that about five percent of the spreadsheets they audited have very serious errors that would have had major ramifications had they not been caught. Audits were done by single auditors, so from the research on spreadsheet and software auditing, it is likely that half or fewer of the errors had been caught. In addition, virtually all of the spreadsheets had standard formats required for their specific legal purposes, so error rates may have been lower than they would be for purpose-built spreadsheet designs.

150 rows contained an error large enough to be financially material. (Sarbanes-Oxley practitioners, please take note.) KPMG, in turn, found that 91% of the spreadsheets created to support decision making had an error large enough to affect the decision.

Why do programming languages and spreadsheet programs create so many errors? As it turns out, they don't. The problem is not with the software but with the people using the software. To put things in a nutshell, many years of human error research in a variety of fields have shown that human beings always have an error rate floor of about 2% to 5% for cognitive actions as complex as those in programming and spreadsheet development. Human error theory suggests that human cognition evolved to balance several competing goals, including accuracy, speed, and the need for rapid changes of attention when needed.

Consider what happened when humans were hunters. If a saber tooth tiger was approaching, a person had to shift attention instantly from what they were doing. This limited the attention given to any task. In addition, if a saber tooth tiger was approaching, it was important to act quickly. A fast action with an occasional probability of error was much better than regret while being chewed on. In addition, while occasional errors do kill hunters, new hunters are easy and fun to make.

Spreadsheet Testing and Inspection

OK, if spreadsheet errors are frequent and important, and if the tools are not to blame, what should spreadsheet developers do? Research has shown that spreadsheet errors look a great deal like software errors in type as well as frequency. This suggests that the whole traditional systems development life cycle (SDLC) discipline in programming will be needed. We must, in effect, teach new dogs very old tricks.

All phases of the systems development life cycle are important, but one is absolutely critical. This is testing. Test-

ing is the *sine qua non* of error reduction. Unless a company spends about a third of the total development effort on testing, nothing else matters.

When programmers talk about testing, they mean something very specific—plugging in input test values and ensuring that the program gives the correct result. However, selecting good test values is extremely difficult to do. In addition, programmers have behavioral “oracles” to tell them what results they should have—for instance, whether a record is updated correctly. However, complex spreadsheets need computational oracles to compute expected results. Most of the time, spreadsheet computations are so complex that there is no alternative but to use a spreadsheet to predict the results. For spreadsheet developers, traditional testing will require extensive training (which is unlikely to be available) and may be prohibitively difficult.

A more attractive approach may be inspection, in which spreadsheet practitioners go over the spreadsheet, module by module, looking at *every* formula cell in the spreadsheet. Limited research indicates that spreadsheet developers are about as good at inspection as professional programmers. Unfortunately, while people are very accurate when they build formulae, their detection rates are much lower. Inspection requires teams of three to five programmers to produce a sufficiently high detection rate. With teams, detection rates rise to between 60% and 80%. With both testing and inspection, there must be multiple rounds at different levels of integration, beginning with unit testing, then testing of small modules.

Conclusion

Why not get rid of spreadsheet programs entirely and replace them with dedicated financial modeling, statistical, and other types of special-purpose software? The most important consideration is that humans will make the same number of errors when logic is involved. Specialized programs should be able to reduce errors by reducing the

number of cognitions users will make, but they will certainly not eliminate errors. Testing and inspection will still be critical. In addition, while spreadsheets are general purpose calculation systems, specialized programs—by definitions—do things that the spreadsheet programs cannot do. If spreadsheets are needed to supplement packaged programs, they often are used for especially complex calculations. ■

EDITOR, from page 3

published by *Business Week* in 1988. Robert Markland of the University of South Carolina compares several major full-time MBA rankings, discusses the pros and cons of the rankings, and examines the methodologies used in the rankings. He suggests that, “All of the stakeholders in the process will gain if we can move away from an obsession on a numerical ranking to a more balanced approach which stresses overall quality.”

Chetan Sankar of Auburn University reviews the book entitled *The Art & Craft of Case Writing* (2nd ed., M.E. Sharpe), by William and Margaret Naumes. The authors drew on their vast experience as case writers in writing this interesting book. The conclusion is that the book will be useful for both novice case writers and those interested in conducting educational case research.

See you all in San Antonio soon! ■

2006 DSI Annual Meeting Website Links

DSI Annual Meeting Homepage
www.dsi-2006.org

Online Conference Registration:
www.decisionsciences.org/CIS

Hotel Reservations
www.stayatmarriott.com/DSI2006/