What We Know About Spreadsheet Errors

Raymond R. Panko
University of Hawaii, USA

Although some spreadsheets are small “scratch pad” applications, many are large and complex, and many mission-critical decisions depend on spreadsheet analyses. In recent years, we have learned a good deal about the errors that people make when they develop spreadsheets. In general, errors seem to occur in a few percent of all cells, meaning that for large spreadsheets, the issue is how many errors there are, not whether an error exists. These error rates, although troubling, are in line with error rates in programming and other human cognitive domains. In programming, we have learned to follow strict development disciplines to eliminate most errors. Surveys of spreadsheet developers indicate that spreadsheet creation, in contrast, is informal, and that few organizations have comprehensive policies for spreadsheet development. Although prescriptive articles have focused on such disciplines as modularization and having assumptions sections, these may be far less important than post-development testing.

Spreadsheet programs are often seen as tools for building small and simple “scratch pad” applications. Yet many spreadsheets are large and complex (Floyd, Walls, & Marr, 1995; Hall, 1996, 78) and many mission-critical organizational decisions are guided by such non-trivial spreadsheets.

Unfortunately, as we will discuss below, there is strong and mounting evidence that spreadsheet errors are widespread. Much of this evidence is available in at least summarized form at the Spreadsheet Research (SSR) website (Panko, 1997b). For instance, this website has data from four systematic field studies involving over 300 operational spreadsheets. It also has data from over a dozen experiments that have used over 1,000 subjects. Every one of these studies, without exception, has found errors at rates that any reasonable person would find to be unacceptable.

The SSR website also contains information from many other studies, including surveys of spreadsheet developers and surveys of organizational control practices for spreadsheet developer. The results show a pattern of considerable care in development yet far less care than professional programmers use when they work, despite strong similarities between spreadsheet and programming errors in both frequency and type.

Error Rates

Introduction

With spreadsheet programs, end users became capable of developing analyses containing thousands of cells. Under these conditions, unless the percentage of incorrect cells is almost zero, there will be a very high probability of bottom-line errors.

Over the years, a number of embarrassing spreadsheet development incidents have been reported (Panko, 1997b). In
most cases, either the firm that made the error was forced to disclose its mistake, or consultants familiar with the error revealed it.

Today, we have moved beyond such anecdotal evidence, into the realm of systematic field audits and laboratory experiments. Table 1 summarizes key data from these studies. Although there is great diversity in methodology and detailed results, one key pattern stands out clearly: every study that has attempted to measure errors has found them and has found them in abundance.

**Consistent with Other Human Error Data**

When most people look at Table 1, their first reaction is that such high error rates are impossible. In fact, they are not only possible. They are entirely consistent with data on human error rates from other work domains. The Human Error Website (Panko, 1997a) presents data from a number of empirical studies. Broadly speaking, when humans do simple mechanical tasks, such as typing, they generally make undetected errors in about 0.5% of all actions. When they do more complex logical activities similar in complexity to spreadsheet development, the error rate generally rises to about 5%. So if we saw lower error rates in either spreadsheet experiments or field audits, we would have to question the results.

The most complete set of data on error rates comes from programming, which is at least a cousin of spreadsheet development and has similar error rates. In programming, many firms practice code inspection on program modules. In code inspection, teams of inspectors first inspect the module individually and then meet as a group to go over the module again (Fagan, 1976). Data from literally thousands of code inspections (Panko, 1997a) converges to an error rate of roughly 5% of all program statements after the developer has finished building and checking the module (Panko, 1997a).

There is even an emerging theory for why we make so many errors (Reason, 1990). In general, the emerging theory argues that human beings are amazingly fast and flexible and can juggle multiple tasks and constraints. However, the same cognitive processes that allow us to work this way contain tricks that inevitably produce occasional errors. Alexander Pope wrote that “To err is human.” Today, we can both explain why this is so and can even quantify it.

**Measuring errors**

The field audit studies shown in Table 1 report a simple measure of spreadsheet errors—the percentage of spreadsheets that contain at least one serious error. (Minor errors are discounted). While this is a vivid statistic, it does not tell us how many errors an average spreadsheet contains.

A better measure is the cell error rate (CER), which is the percentage of numerical and formula cells that contain errors. If experience with programming is correct, CER should be roughly independent of spreadsheet size, although not perfectly so. In other words, if we know the size of a spreadsheet, we can use a reasonable CER to estimate how many errors the spreadsheet probably has.

**Errors by Life Cycle Stage**

In programming, error rates vary by life cycle stage. Programmers make many errors when they are building a module but catch many of these errors before they finish the module and submit it to code inspection, data testing, or both. Module code inspection and testing catch more errors, and a second wave of code inspection and testing at final assembly catches still other errors. Relatively few errors should survive into the final operational systems.

**Errors During Cell Entry**

Table 1 indicates that only two studies have looked at errors during cell entry (Lerch, 1988; Olson & Nilsen, 1987-1988), that is, when the developer is entering formulas. In these studies, errors were counted as they were made, so many of the errors counted would be corrected spontaneously by the developer. Still, they show that people do make many errors when they are working on spreadsheets.

**Errors at the End of the Development Stage**

Most studies in Table 1 looked at errors at the end of the development stage, when subjects said that they were finished developing their spreadsheets. Apart from research that looked only at selected high-risk formulas (Janvrin & Morrison, 1996), cell error rates across these studies were similar, despite the fact that subjects ranged from novices to highly experienced spreadsheet developers. One study (Panko & Sprague, Forthcoming) directly compared undergraduate business students, MBA students with little spreadsheet developing experience, and MBA students with more than 250 hours of spreadsheet development experience. Their CERs were almost identical. Even when a task was selected to be very simple and almost completely free of domain knowledge requirements (Panko & Sprague, Forthcoming; Teo & Tan, 1997), about 40% of all spreadsheets contained errors, and the CER was about 2%.

**Errors Found in Code Inspection**

Code inspection or some other form of intensive testing may be needed to detect errors that remain at the end of the development stage. Fagan (1976) developed a comprehensive discipline for code inspection. As noted earlier, this method has a group of individuals inspect a program module individually by going through the module line by line. Then, the team meets as a group and again goes through the module line by line.

Code inspection is very difficult. In programming experiments, it has been found that individual inspectors catch only half or fewer of all errors in program modules (Panko, 1997a). Even real-world group inspection usually only catches 80% or fewer of all errors in a program module (Panko, 1997a).

In code inspection, as in development, spreadsheeting
Related Content

An Extension of the Technology Acceptance Model to Determine the Intention to Use Biometric Devices
[www.igi-global.com/chapter/extension-technology-acceptance-model-determine/18153?camid=4v1a](www.igi-global.com/chapter/extension-technology-acceptance-model-determine/18153?camid=4v1a)

Design and Implementation of Binary Tree Based Proactive Routing Protocols for Large MANETS
[www.igi-global.com/chapter/design-implementation-binary-tree-based/73222?camid=4v1a](www.igi-global.com/chapter/design-implementation-binary-tree-based/73222?camid=4v1a)

Microcomputer Software Evaluation and Selection Strategies
[www.igi-global.com/article/microcomputer-software-evaluation-selection-strategies/55647?camid=4v1a](www.igi-global.com/article/microcomputer-software-evaluation-selection-strategies/55647?camid=4v1a)

Human Systems Engineering and Educational Technology
[www.igi-global.com/chapter/human-systems-engineering-and-educational-technology/183010?camid=4v1a](www.igi-global.com/chapter/human-systems-engineering-and-educational-technology/183010?camid=4v1a)