ERDC TN-EMRRP-EBA-03
September 2009

Reducing Spreadsheet Errors

by S. Kyle McKay[1]

OVERVIEW: Increasing computational power and memory capacity in desktop computers has resulted in access to numerical techniques previously unavailable to many users. Although computational capabilities have increased, many users are unable to take full advantage of this power due to limited programming expertise. Spreadsheets offer these users a means to capitalize upon computational potential without extensive training, though spreadsheet models can confer risks due to high error rates and lack of quality control. Table 1 presents several strengths and weaknesses associated with using spreadsheets as numerical tools for the masses.

Table 1. Strengths and shortcomings of spreadsheets as computational tools.

Strengths	Shortcomings
• Inexpensive to develop • Readily available • Transferable to a broad user base • Simple, logical interfaces easily understood and applicable without extensive training • Built-in tools and third-party add-ins provide computational power without extensive programming • Current spreadsheet programs have many capabilities previously associated with more sophisticated software (e.g. optimization routines)	• False sense of confidence due to the easy output of "answers" • Can quickly become complex • Error rates are high • Troubleshooting and error checking may be time-consuming and difficult (as is the case with all end-user computer programs) • Written quality assurance and control techniques are not often applied due to reliance on self-policing

Both researchers and practitioners have cautioned users about high rates and consequences of errors in spreadsheets. Computer programming and finance literature have shown that, regardless of complexity, errors occur in approximately 90 percent of spreadsheets (Panko 2005; Powell et al. 2008b) and 1-5 percent of cells (Panko and Sprague 1998; Powell et al. 2009). People typically commit undetected errors in approximately 0.5 percent of simple mechanical tasks (e.g., typing) and error rates increase with complex logical tasks (e.g., programming; Panko 2005). Consequently, if 5 percent of cells contain errors, then complex spreadsheets with thousands of cells are likely to contain mistakes. However, amongst error rate studies there is not a clear definition of what constitutes an error or how overall results are impacted by errors (Powell et al. 2008b).

Computational errors are not the only errors possible in spreadsheets, and many errors lie outside the realm of what model developers can control, including: data errors (e.g., uncalibrated instrument), input errors, user errors (e.g., manipulation of spreadsheet structure through data sorting), appropriateness of application to the problem, reliance on conceptual or technical

[1] Research Civil Engineer, U.S. Army Engineer Research and Development Center, Environmental Laboratory, Athens, GA. *Kyle.McKay@usace.army.mil*, 601-415-7160.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
SEP 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Reducing Spreadsheet Errors	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Army Engineer Research and Development Center,Environmental Laboratory,3909 Halls Ferry Road,Vicksburg,MS,39180-6199**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**13**	

information that may be uncertain or erroneous (e.g., sea level rise predictions), misinterpretation of results, and deliberate errors associated with fraud.

U.S. Army Corps of Engineers (USACE) Engineer Circular 1105-2-407 requires that all models used for planning be peer reviewed and certified (USACE 2005). Under this policy, a model is defined as "a representation of a system for a purpose" (USACE 2007). Within USACE, spreadsheets are often used to represent systems and inform decisions ranging from dredging schedules to restoration outcomes to budget allocation. These tools range in complexity from simple data storage to intricate models of complex system properties depending on an array of inputs, conditional arguments, and possibly third party add-ins.

Although spreadsheets are ubiquitous, high error rates suggest that thorough checking, testing, and auditing are uncommon. This paper provides guidance for quality assurance and quality control practices and techniques for avoiding or reducing errors by: 1) planning spreadsheet development, 2) avoiding errors in development, 3) finding errors, and 4) self-improvement. This document is not intended to be a hard and fast set of "rules," but rather to provide guidance for spreadsheet developers. This technical note focuses on the USACE standard spreadsheet program, Microsoft® Excel 2003.

PLANNING SPREADSHEET DEVELOPMENT: Although the complexity and consequence of a given spreadsheet should dictate the amount of time, resources, and energy invested, even the simplest tools stand to be improved through planning prior to development (Kruck 2006).

Although extensive guidance on spreadsheet development exists, the basic properties of a well-designed spreadsheet as listed below are relatively few (Read and Batson 1999; Bewig 2005), and spreadsheet developers, reviewers, and users all benefit from understanding and applying these properties throughout a spreadsheet's life:

- *Simple* – Following Einstein's guidance, a spreadsheet should be designed to be as simple as possible, but no simpler. This concept extends not only to design, but also to ease of use, understandability, and clarity of the spreadsheet and outputs.
- *Relevant* – Spreadsheets are developed for a specific purpose. If this purpose is not met or the relevant issues are not addressed, then the spreadsheet is meaningless. As such, a robust spreadsheet focuses on the key questions of concern and does not distract the user with extraneous information.
- *Adaptable* – Even after extensive planning, the first draft of a spreadsheet will likely fall short of needed capability; designers need to plan for multiple iterations. Design proactively to anticipate change by creating an adaptable spreadsheet with a simple, tractable design that can be followed and modified without inadvertently altering calculations.
- *Reliable* – A robust spreadsheet is a well-tested tool proven time and again to produce accurate results for a given input set.
- *Appropriate* – A spreadsheet is not the tool for every job. Spreadsheet programs contain numerous capabilities and functions; however, other programs may be more appropriate for a given situation. Appropriate use of tools internal and external to a spreadsheet should be considered early in development.

Spreadsheet development should be an iterative and collaborative process, and understanding a spreadsheet's audience and life cycle may aid in planning spreadsheet development. The audience and user community for a spreadsheet may include the *sponsor*, who assigned the task; the *developer* (you), who designs the spreadsheet; the *auditor*, who checks the validity of the spreadsheet; the *user*, who inputs data and operates the model; and the *reader*, who uses the results of the model to make informed decisions (Read and Batson 1999; Bewig 2005). In spreadsheet programming, developers, auditors, users, and readers of a model are often the same individual or team and many errors are associated with a lack of collaboration and thorough review.

Understanding a spreadsheet's life cycle may lend perspective to the development process (Janvrin and Morrison 2000). Because no model moves from concept to implementation without flaw, this process is iterative and includes the following six steps (from Read and Batson 1999, Figure 1):

- *Scope* – "Think before you write" (Bewig 2005). Scoping includes making decisions regarding model structure and function, key processes and variables to include (or exclude), needed resolution, and scale (Kruck 2006). This process is both analogous to and applies products from conceptual modeling exercises.
- *Specify* – Specify the logic of the model in sufficient detail to guide design of the spreadsheet (O'Beirne 2005).
- *Design* – Develop the structure of the spreadsheet. (NOTE: Diagramming design on a whiteboard is often helpful.)
- *Build* – Use the spreadsheet software to construct or code the model. Lay out the spreadsheet based on the needs of the problem at hand (Kruck 2006).
- *Test* – Check the spreadsheet for errors (conceptual as well as numerical), and verify and validate results.
- *Use* – Apply the model to answer questions and inform decisions.

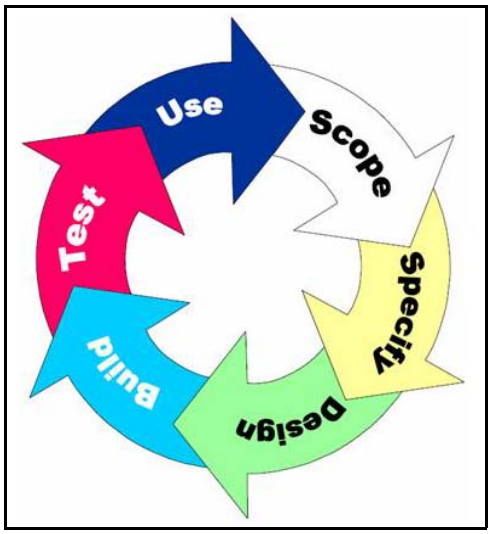

Figure 1. Stages of a spreadsheet's life cycle (from Read and Batson 1999).

AVOIDING ERRORS DURING DEVELOPMENT: Listed below are 25 suggested practices for reducing spreadsheet errors compiled from literature, the author's experience, and discussions with colleagues. For easy use, best practices have been broken into categories. Notably, many of these techniques are not universally recommended and may be in opposition to one another. As such, their use should be considered on a case-by-case basis.

General

1) Work in groups (Bewig 2005). Panko and Halverson (2001) demonstrated that working in groups of three rather than alone reduced cell error rates by 78 percent!

2) Build complex spreadsheets in stages and test results as you go (Bewig 2005).

3) Do not just document problems, fix them immediately. Also, avoid "quick fixes" such as replacing a formula with a constant to validate outputs (Bewig 2005).

4) Take advantage of Visual Basic for Applications (VBA) to improve spreadsheets. Programming and coding portions of a spreadsheet in VBA (especially iteration) can reduce effort as well as errors. Users unfamiliar with VBA may begin learning by "recording macros" in Excel. Microsoft's online tutorials (*www.office.microsoft.com/en-us/excel*) provide overviews of this and other VBA capabilities.

5) Thorough documentation of spreadsheet development and application is critical. Documentation should minimally specify purpose, assumptions, and logic of the spreadsheet; record the development process; explain operation of the spreadsheet; and explain update procedures. Documentation may take many forms; the "best" form depends upon the purpose and scope of the spreadsheet. A combination of the following techniques will help record development history, inform users, and expedite the review process.

   a) "HOWTO" Tab – A separate worksheet explaining logic, noting assumptions, stating spreadsheet version (e.g., "1.0" or 20 January 2009), developer contact information, user instructions, and other information provides developers and users a readily available reference (Bewig 2005; O'Beirne 2005; Pryor 2006).
   b) In-cell comments – Using the reviewing toolbar, developers can insert comments that provide information on ranges, references to data sources, and logic of formulas for individual cells (Bewig 2005; O'Beirne 2005; Figure 2).
   c) Internal documentation – Range descriptions in nearby cells, text boxes, and code development comments (in VBA) are also common forms of documentation well-suited to instruction and explanation, particularly for less experienced users, due to their close proximity to the cell in question (Pryor 2006; Figure 2). Extensive use of these comments may, however, overcomplicate a spreadsheet and distract users.

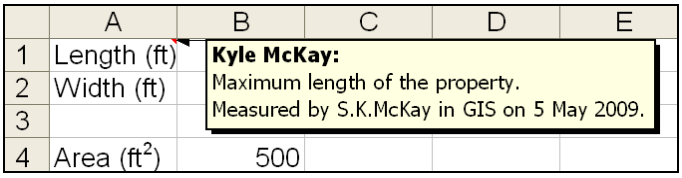

Figure 2. Example of in-cell comments and internal documentation.

   d) External Documentation – For complex spreadsheets, documentation outside of the spreadsheet tracks relevant information and provides spreadsheet users and readers additional guidance. If external documentation is produced, effort should be made to maintain consistency between documentation and model versions (Pryor 2006).

6) Descriptive naming of cells, rows, columns, worksheets, and workbooks clarifies intent and purpose of the object (Bewig 2005; Kruck 2006).

   a) In the sciences, cells are often referred to by variables, though individuals or specific disciplines may use the same variable to represent a different concept. For example, *n* is often used to express sample size in statistical analyses; however, in river science *n* commonly refers to Manning's roughness coefficient.

b) Workbook (file) naming should succinctly indicate the purpose and version of a spreadsheet (e.g., 'USACE_Budget_2009-01-20.xls'). Many computational programs (including Excel and VBA) prefer continuous worksheet (tab) and workbook (file) names; thus, avoid spaces in naming to prevent referencing errors.
c) Label dimensional quantities with units (e.g., $, miles). Verify all units and unit conversions in the spreadsheet.

7) Use reference styles and locations correctly.

a) Confusion and misapplication of cell referencing (e.g. A1, A1, R1C1) is a common source of spreadsheet errors (Powell et al. 2009). Some authors advocate use of particular referencing styles; however, if applied correctly, each style can produce accurate outputs (O'Beirne 2005; Kruck 2006). A thorough understanding of differences in absolute (A1), relative (A1), and mixed (A$1) references is critical because each is impacted differently by insertion, deletion, copy/paste, sorting, and other movement commands (Bewig 2005).
b) Some referencing errors may be reduced through use of range names, which provide users and reviewers with easy-to-read formulas. For instance, if area is calculated in cell B4 using length and width (B1 and B2, respectively), the use of range names for length and width provides the formula "=Length*Width" rather than the less intuitive "=B1*B2".
c) Use caution when referencing cells in other worksheets and workbooks (Bewig 2005). Reviewing formulas and verifying that the correct cells are referenced is very challenging when cells are on different worksheets. As such, use links to other worksheets and workbooks sparingly (if at all). These references create an extremely high potential for formula errors or links to spreadsheets that may contain errors and can induce directory and updating problems (O'Beirne 2005).
d) Users should be sure that they understand functions used to find or look up data before applying them (e.g., VLOOKUP, MATCH). These functions create complex, albeit useful, statements that are error prone (O'Beirne 2005).

8) Use security features appropriately to increase reliability (Bewig 2005). Locking or protecting non-input cells (and releasing the key) is a good way to maintain consistency and avoid unintentional changes. However, locked spreadsheets are difficult to review and update as errors are discovered. O'Beirne (2005) suggests locking a spreadsheet using a blank password to protect against unintentional errors, but to otherwise allow modification.

9) Delete extraneous information and cells that detract from the message and purpose at hand. Cells without dependents and duplicate cells (A4=A2) are candidates for removal (Raffensperger 2001).

10) Be forewarned that copy/paste and autofill errors are common (Bewig 2005; Powell et al. 2009). For instance, changing some but not all cells in a range and creating incomplete ranges are common but easily avoidable errors. Also, copy/pasting a worksheet truncates any cell with more than 255 characters, so large passages of text may not transfer completely.

Logic and Calculation

11) Complicated expressions are potentially hazardous in terms of error risk. Experts warn users of this error source, but there is significant debate as to whether it is better to break up complex arguments into multiple expressions (Bewig 2005; O'Beirne 2005; Kruck 2006) or be concise and minimize the total number of cells (Raffensperger 2001). A third alternative is to follow both procedures and use this as an opportunity to verify correctness of results.

12) Avoid "hard-wiring" constants into formulas as this reduces adaptability (e.g., unit conversions or constants; O'Beirne 2005; Kruck 2006). Consider using separate cells for unit conversions, discount rates, and other constants so auditors can more easily track and verify assumptions (Bewig 2005).

13) Maintain a common interval for calculation (e.g., one year per row); this can help users and readers understand a spreadsheet (Raffensperger 2001; Bewig 2005).

14) Arranging inputs close to calculation creates short distances of precedence and provides auditors with a straightforward view of calculations (Raffensperger 2001; Figure 3 below).

15) Reserve circular references as error detection tools (O'Beirne 2005). Learn to apply Goal-Seek and Solver rather than circular references (Bewig 2005). WARNING: Optimization routines such as Solver are often dependent upon the starting value of the parameter being optimized, so test starting values to ensure consistent solutions.

Formatting and Labeling

16) Ensure that a spreadsheet reads like a document, top to bottom and left to right. This will not only increase readability, but will also assist in auditing because dependents will flow in one direction (Bewig 2005; Raffensperger 2001; Figure 3 below).

17) Spreadsheets are simple to understand because they provide structure for numerical problems. As part of the visual structure, the grid is a useful reference in reading spreadsheets. Turning the grid off can disorient users and is not recommended (Raffensperger 2001).

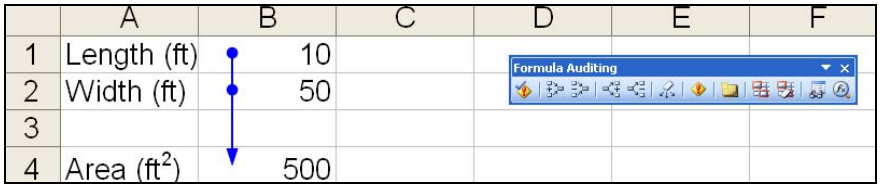

Figure 3. Tracing precedence using the Formula Auditing toolbar.

18) Although color may increase aesthetic appeal and facilitate transfer to non-technical audiences (e.g., public stakeholders), in terms of error reduction, format for function, not for decoration (O'Beirne 2005; Raffensperger 2001). Formatting is particularly useful for identifying cell function. Different format (background color, border, font, etc.) for input, calculation, and output cells allows users to quickly find cells of a specific type (Raffensperger 2001; Bewig 2005; Figure 4 below). Try to make the system as logical as possible (e.g.,

input = green background, calculation = yellow background, error = red background, output = larger, bold font) and include the key within the spreadsheet and/or documentation. While no agreement exists on the use of color, in general, minimize color in spreadsheets because it is often lost in printing. Raffensperger (2001) suggests limiting yourself to three colors that are distinguishably different when printed. Be aware of color-blind team members and readability on the computer – some users find that bright colors tire their eyes.

19) Organize the spreadsheet based on functionality (e.g. input, calculation, output; O'Beirne 2005). There is debate over the use of multiple spreadsheets to separate these functions; however, in general, use additional worksheets sparingly (Bewig 2005; Raffensperger 2001). This will allow auditors and users to easily find information and understand logic. Also, delete blank worksheets (Raffensperger 2001; O'Beirne 2005).

	A	B	C	D	E
1	Length (ft)	10		Input	
2	Width (ft)	50		**Calculation /Output**	
3					
4	Area (ft^2)	**500**			

Figure 4. Example of formatting for function.

20) Remember significant figures and set cell format to show appropriate levels of precision. Also, be aware of the impacts of rounding values (O'Beirne 2005).

21) Make blank cells look blank. Do not hide active cells by formatting text as white, covering them with figures, using an if-then statement, or by using spaces or punctuation; this will reduce omission errors, such as missing input values (Raffensperger 2001; O'Beirne 2005).

Figures. Although not explicitly part of spreadsheet computations, figures and charts may present results to users and readers in ways that obscure findings, implicitly creating errors.

22) Use the right chart for the job (Bewig 2005). Bar graphs, x-y scatter plots, line graphs, and pie charts do not all show the same information; understand and use them appropriately.

23) Consider changing chart location. Charts embedded in worksheets provide users quick access, but may be distorted or can obscure cells. By creating a new tab for important charts, formatting becomes more consistent and results more readable.

24) Modify charts such that all data sets are visible (e.g., orientation of a 3D chart may obscure a data set; O'Beirne 2005).

25) Charts may be more effective or clear if the scale is adjusted to the data set; however, if calculations change outputs, then a specific scale adjustment may no longer be appropriate (O'Beirne 2005).

FINDING ERRORS: Testing is the most important phase of spreadsheet development. Testing is not simply a single test run or a quick review by a colleague, but rather should be a comprehensive process that overlaps with design and construction of the spreadsheet. Panko (2007) suggested that testing and auditing should comprise approximately one third of total development time. Spreadsheet auditing should be done by the developer as well as peer reviewers

(developer-specified as well as blind). This section introduces testing and auditing techniques that both developers and peer reviewers should apply.

Testing and auditing techniques can be applied with varying success based on how rigorously the audit is conducted. With increasingly thorough reviews and greater numbers of reviewers comes greater reduction in errors and increasing confidence in the spreadsheet. Panko (2007) highlighted the importance of group work at this stage of spreadsheet development, stating, "When programmers inspect code, they typically find 30 percent to 50 percent of all errors. This is why programmers do inspection in teams. Individual [programmers] only catch about 60 percent of all errors, and this goes up to about 80 percent in teams." However, regardless of the number of reviewers, all errors may not be detected and reviewers can only state their professional opinion that the spreadsheet provides appropriate results (Pryor 2004). Finally, reviewers must be cautious not to induce errors as part of the review process through erroneous "correction" of cells or unintentional changing of cells (Teo and Tan 1999).

Review during spreadsheet construction. Developers should inspect cells as equations are entered. Additionally, review of preliminary versions can verify scope and structure of the spreadsheet, provide input that may be incorporated in development, and catch early errors before they cascade through the spreadsheet.

Use cross-checks and validation points. Using "sanity checks" that verify the reasonableness of outputs can save time and effort on the part of the developer and reviewer. For instance, probability distributions should be summed to verify that cumulative distributions total 100 percent. Other techniques for cross-checking results include, but are not limited to, summing across rows and columns, applying different calculation methods to the same problem, applying conditional formatting to identify inputs, intermediate steps, and results outside of expected ranges (e.g., habitat units less than zero), and using IF statements to point out calculation flaws (Bewig 2005; O'Beirne 2005).

Test spreadsheets. Pryor (2004) distinguished between reviewing models by examining code and formulas for errors versus actually running the models and examining results. When testing spreadsheets, multiple inputs covering expected ranges should be used to verify the model. Some particularly useful input conditions include typical values with known results, boundary conditions (e.g., if river discharge is zero, habitat for aquatic obligates should be zero), values intended to "break" the model, tests of logical statements (e.g., correct results should be provided for switches in IF statements and lookup functions), and step-wise values intended to verify one portion of the model (Pryor 2004). A number of authors encourage documenting test cases and results as part of the development and review process (Pryor 2004; O'Beirne 2005; Panko 2007). Although tests may provide interested parties with confidence in the spreadsheet, testing results should be viewed with caution based on the source of verification values. For instance, when verifying results against a previous, error-filled version of the model, tests are not verifying accuracy, but consistency (Teo and Tan 1999). At best, full spreadsheet testing merely verifies results and does not provide the location of specific errors.

Inspect all formulas. Comprehensive cell-by-cell examination of all inputs and formulas is recommended for all spreadsheets. However, for even moderate-sized spreadsheets, this process is very time-consuming and reviewers may fatigue and become less thorough before the task is

finished (O'Beirne 2005). Although some of the simplest calculations are often performed early in computation, errors in these simple calculations have dramatic effects because they cascade through the spreadsheet (Bewig 2005). Check all formulas, not just complex ones. To facilitate this review, Excel provides developers and reviewers with tools to view and print cell formulas.

Use auditing tools. Considering the complexity often encountered in spreadsheets and concomitant difficulty of testing and review, thorough spreadsheet testing and cell-by-cell auditing is rarely used. To facilitate more rapid audits and highlight error-prone portions of the spreadsheet, a number of auditing tools have been developed and are quite useful (O'Beirne 2005; Powell et al. 2008a). These tools should be applied as a secondary review of the spreadsheet even if formula auditing has occurred. The following is an abbreviated list of available auditing tools to consider applying in spreadsheet review. Many of these tools are free to download (*indicates current ACE-IT approval).

	A	B	C	D
1	Length (ft)	10		Input
2	Width (ft)	50		Calculation /Output
3				
4	Area (ft^2)	=B1*B2		
5	Perimeter (ft)	=2*B1+2*B2		

Figure 5. Example of formula viewing capabilities (in Excel 2003 this is found at Tools > Options > View tab > Window Options > Formulas).

- Microsoft Excel Built-In Auditing Tools*
  - "Formula Auditing" toolbar (see Figure 3)
  - Warning messages – circular references are identified, suspicious cells are flagged with small green triangles, and error messages (e.g., #VALUE!) identify incorrect or invalid cells
- XL Analyst – *http://www.xlanalyst.co.uk*
- Spreadsheet Professional – *www.spreadsheetinnovations.com*
- Operis Analysis Kit – *www.operis.com/oak.htm*
- Spreadsheet Detective – *http://www.spreadsheetdetective.com/*
- XLSior – *www.xlsior.com*

The following tools may be applied to compare different versions of a spreadsheet:

- Synkronizer – *http://www.synkronizer.com*
- Florencesoft DiffEngineX – *http://www.florencesoft.com/*

Be cognizant of other error sources. When auditing for errors, focus on computational errors, but audits also provide an opportunity to identify additional errors such as input errors, user errors (e.g., manipulation of spreadsheet structure through data sorting), and deliberate errors associated with fraud.

SELF-IMPROVEMENT: In addition to the development, avoidance, and auditing techniques discussed above, spreadsheet developers, users, and readers can reduce error rates by improving themselves, their level of spreadsheet knowledge, and corporate policies.

Be humble. Panko (2005) reviewed a number of spreadsheet confidence studies and found that although spreadsheets reflected typical error rates, users were consistently overconfident in their

models. Curiously, confidence has been shown to increase with degree of difficulty, although the probability of errors in complex models is higher due to greater cell numbers. Panko (2005) also noted that although experienced programmers were more likely to take advantage of controls, the difference in error rates associated with experience was only "modest." Furthermore, because developers do not often formally test spreadsheets, when they do stumble upon errors they become overconfident in their ability to find and eliminate errors. In general, overconfidence of both novice and experienced professionals is generally unwarranted and users should be more cognizant and cautious of their shortcomings (Panko 1999).

Be consistent. Consistency in technique, documentation, and formatting makes it easier to pick up old models and understand the logic, structure, and interface (O'Beirne 2005).

Do not get caught by avoidable problems. Failing to save work or saving over an earlier (audited) version of a spreadsheet are examples of simple, avoidable mistakes. Turn on backup features such as auto-recover and periodically back up work externally (Do not keep the backup in the same place as the original!). Save files explicitly. When making substantive changes, do not risk saving over a version that may be needed later; save a new version.

Stay organized. Know where files are stored and be able to access them readily. During the development phase, users may use challenging functions they have worked through before. Do not duplicate effort; look back at old work. This is especially useful when working with VBA code.

Be a student of spreadsheets. Expose yourself to spreadsheets. More educated spreadsheet readers become better spreadsheet developers by learning best practices and techniques through the work of others. These developers learn from their peers and take advantage of their knowledge. Do not duplicate time someone else has invested in learning a technique.

Use available resources. Countless books have been written regarding every aspect of spreadsheet practice from error checking (O'Beirne 2005) to numerical techniques (Billo 2007). Many technologies are forthcoming that are helping spreadsheet developers become more efficient and accurate (e.g., free auditing software). Microsoft, professional organizations, and spreadsheet researchers maintain many useful websites. A few prominent websites include:

- Microsoft Office Online, *http://office.microsoft.com/en-us/excel/default.aspx*
- European Spreadsheet Interest Group (EUSPRIG), *www.eusprig.com*.
- John Walkenbach, Spreadsheet Expert, *http://spreadsheetpage.com/*
- Ray Panko, University of Hawaii, *http://panko.shidler.hawaii.edu/*
- John Raffensperger, University of Canterbury, *http://www.mang.canterbury.ac.nz/people/jfraffen/*
- Stephen Powell, Dartmouth College, *http://mba.tuck.dartmouth.edu/pages/faculty/steve.powell/*
- Systems Modeling Ltd., *http://www.sysmod.com/*
- Richard Brenner, *http://chacocanyon.com/smm/*

Develop corporate policy. Although spreadsheet error rates and developer overconfidence are well documented, in general, spreadsheets are not acknowledged as regular sources of poten-

tial error by scientists and engineers. In contrast, following well-publicized accounting scandals (e.g., Enron), Congress developed legislation to exert controls on accounting practices that include spreadsheets, namely the Sarbanes-Oxley Act of 2002 §404. As such, accounting and finance firms have acknowledged the likelihood of errors and corporations have begun instituting controls to minimize them. Although fields of science and engineering have not yet reached this level of quality assurance in spreadsheet practices, the Corps of Engineers has taken steps to avoid costly mistakes through model certification (USACE 2005, 2007). Under this purview, planning models must be certified for conceptual and technical soundness as well as computational correctness. Additionally, the following guidelines may be considered:

- Spreadsheet development and auditing controls should be risk-based. Level and type of control can be based on frequency of spreadsheet use and the consequences of inaccurate results. Unfortunately, consequences of spreadsheet errors are one of the least studied aspects of spreadsheet error research (Powell et al. 2008b).

- Regardless of institutional certification requirements, due to the ubiquity and utility of spreadsheets, spreadsheet developers/users must at some level be self-policing and self-reporting in spreadsheet development and auditing. Thus, training spreadsheet developers, users, and readers in potential hazards, avoidance, and self-audit may help create a corporate culture of quality assurance.

- Modeler accountability must be addressed. High levels of developer accountability and penalization for errors may discourage use of spreadsheets altogether or decrease admission of errors (Panko 2005). Conversely, developer immunity may not support error avoidance and thorough spreadsheet auditing.

- According to Powell et al. (2008a), "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." Select USACE personnel could be trained explicitly to audit, test, and edit spreadsheets to adhere with model certification requirements and/or industry standards.

CONCLUSIONS: Spreadsheets are powerful tools with countless potential applications. As with any computational tool, errors have been shown to be prevalent (Panko 2005). This document has presented a number of techniques for minimizing and avoiding errors associated with spreadsheets. Techniques have been presented through four major error reduction practices: 1) planning spreadsheet development, 2) avoiding errors during development, 3) finding errors, and 4) self-improvement. By following these (and other) practices, users can become confident in the accuracy of their spreadsheets and the validity of decisions resulting from them. Bewig (2005) summarized the spreadsheet dilemma well, stating:

> *It's good to ask 'Is your spreadsheet right?' but confidence comes from answering the question 'How do you know your spreadsheet is right?' That's a really tough question. If all you can say is that you're a Spreadsheet Super Man, you should hang up your blue tights and red cape and stop writing spreadsheets. But it would be most impressive if you could say:*

> *'I followed accepted best-practice in developing my spreadsheet. I designed my work carefully, and reviewed it when I was finished, using a commercial auditing tool. I tested the spreadsheet using known results, according to a written test plan, and compared to other similar work. Knowing that the result was important, I had Bob, Mary and Joe peer-review my work. There could still be mistakes, but I was at great pains to prevent them.'*
>
> *What's your answer?*

ACKNOWLEDGEMENTS: Research presented in this technical note was developed under the Environmental Benefits Analysis (EBA) Research Program. The USACE Proponent for the EBA Program is Rennie Sherman and the Technical Director is Dr. Al Cofrancesco. Technical reviews were provided by Sarah Miller, Kelly Burks-Copes, and Dr. Craig Fischenich (ERDC Environmental Laboratory), and Virgil Langdon (Huntington District). The use of Microsoft products does not represent an endorsement of these products by either the author or the Department of the Army. (Microsoft, Windows, Excel, Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries." ©2003 Microsoft Corporation. All rights reserved.)

ADDITIONAL INFORMATION: For additional information, contact the author, S. Kyle McKay (601-415-7160, *Kyle.McKay@usace.army.mil*), or the manager of the Environmental Benefits Assessment (EBA) Research Program, Glenn Rhett (601-634-3717, *Glenn.G.Rhett@usace.army.mil*). This technical note should be cited as follows:

> McKay, S. K. 2009. *Reducing spreadsheet errors.* EBA Technical Notes Collection. ERDC TN-EMRRP-EBA-03. Vicksburg, MS: U.S. Army Engineer Research and Development Center. *http://cw-environment.usace.army.mil/eba/*

REFERENCES

Bewig, P. L. 2005. How do you know your Spreadsheet is Right? Principles, Techniques, and Practice of Spreadsheet Style. *www.eusprig.org*.

Billo, E. J. 2007. *Excel for scientists and engineers: Numerical methods*. Hoboken, NJ: John Wiley & Sons, Inc.

Janvrin, D., and J. Morrison. 2000. Using a structured design approach to reduce risks in end user spreadsheet development. *Information & Management*. 37 (1): 1-12.

Kruck, S. E. 2006. Testing spreadsheet accuracy theory. *Information and Software Technology*. 48 (3): 204-213.

O'Beirne, P. 2005. Spreadsheet Check and Control: 47 Key Practices to Detect and Prevent Errors. Wexford, Ireland: Systems Publishing.

Panko, R. R. 1999. Applying code inspection to spreadsheet testing. *J. of Mngmt. Info. Sys.* 16 (2): 159-176.

Panko, R. R. 2005. What We Know About Spreadsheet Errors. *http://panko.cba.hawaii.edu*

Panko, R. R. 2007. A Rant on the Lousy Use of Science in Best Practice Recommendations for Spreadsheet Development, Testing, and Inspection. *http://panko.cba.hawaii.edu*

Panko, R. R., and R. P. Halverson. 2001. An experiment in collaborative development to reduce spreadsheet errors." *Journal of the Association for Information Systems* 2 (4).

Panko, R. R., and R. H. Sprague. 1998. Hitting the wall: Errors in developing and code inspecting a 'simple' spreadsheet model. *Decision Support Systems* 22: 337-353.

Powell, S. G., K. R. Baker, and B. Lawson. 2008a. An auditing protocol for spreadsheet models. *Information and Management* 45: 312-320.

Powell, S. G., K. R. Baker, and B. Lawson. 2008b. A critical review of the literature of spreadsheet errors. *Decision Support Systems* 46: 128-138.

Powell, S. G., K. R. Baker, and B. Lawson. 2009. Errors in operational spreadsheets: A review of the state of the art. In *Hawaii International Conference on System Sciences, January 2009.*

Pryor, L. 2004. When, why, and how to test spreadsheets. In *Proceedings of the Eusprig – 2004.* Klagenfurt, Austria: European Spreadsheet Risk Interest Group.

Pryor, L. 2006. What's the Point of Documentation? *www.louisepryor.com*. Accessed 20 January 2009.

Raffensperger, J. 2001. SpreadsheetStyle.com: When it has to be right. *http://www.mang.canterbury.ac.nz/people/jfraffen/*. Accessed 20 January 2009.

Read, N., and J. Batson. 1999. *Spreadsheet modeling best practice*. Institute of Chartered Accountants.

Teo, T. S. H., and M. T. Tan. 1999. Spreadsheet development and 'what-if' analysis: Quantitative versus qualitative errors. *Accounting Management and Information Technologies* 9: 141-160.

U.S. Army Corps of Engineers (USACE). 2005. *Planning Models Improvement Program: Model certification*. Engineer Circular 1105-2-407, 31 May 2005, Washington, DC.

U.S. Army Corps of Engineers (USACE). 2007. *Protocols for certification of planning models.* Washington, DC: Planning Models Improvement Program.