The EUSES Spreadsheet Corpus: A Shared Resource for Supporting Experimentation with Spreadsheet Dependability Mechanisms*

Marc Fisher II and Gregg Rothermel†
Department of Computer Science and Engineering
University of Nebraska-Lincoln
{mfisher,grother}@cse.unl.edu

December 6, 2004

1 Introduction

Recently there has been a great deal of interest in the domain of spreadsheet programming, with many ongoing projects attempting to provide tools and methodologies to help users create more dependable spreadsheets. These projects have used various methods for evaluating their effectiveness. Rothermel, et al. have developed the WYSIWYT (What You See Is What You Test) family of methodologies that provide immediate incremental feedback to assist users with testing and debugging spreadsheets [6, 7, 9, 10, 12, 13, 14, 15], and have evaluated these methodologies within the context of Forms/3 with a variety of spreadsheets they have created, some based on spreadsheets created by Excel users. Ayalew, et al. have developed testing and debugging methodologies in Excel [4, 5], but have not published results evaluating their methodologies' effectiveness. Several researchers have developed unit inference mechanisms for Excel [1, 2, 3]. These techniques have been validated against a small collection of spreadsheets from a book on using spreadsheets in science and engineering [8], and, in one case, against a small number of spreadsheets created by students in an introductory computer science course.

The studies just described have yielded some understanding of the methodologies studies; however, none of these studies has performed validation against a wide range of spreadsheets in use by real users. There are many possible reasons why such validation has not occurred, but one likely reason is that assembling a large collection of "real-world" spreadsheets is an expensive task. Therefore, we have undertaken the task of assembling and maintaining a corpus of spreadsheets suitable for evaluating these wide-ranging methodologies for helping end users develop dependable spreadsheets.

*University of Nebraska-Lincoln Computer Science and Engineering Technical Report TR04-12-03
†Portions of this work were carried out while the authors were at Oregon State University

2 Spreadsheets

Google Keyword	Initial	Unuseable	Duplicates	Final
database	904	59	125	720
financial	902	31	91	780
grades	895	17	148	731
homework	950	29	239	682
inventory	891	49	86	756
modeling	966	51	183	732
Forms/3	26	0	0	26
From [8]	45	0	0	45
From [11]	13	0	0	13
OSU CS101	9	1	0	8
Personal	5	0	0	5
Totals	5606	236	872	4498

Table 1: Source of spreadsheets surveyed

To obtain a large sample of spreadsheets created by a wide variety of users, we used a Java-based interface to the Google search engine. This interface lets users specify search phrases and filetypes, along with a number of desired results. The engine then retrieves results, attempting to filter out duplicates, as a series of urls appropriate for use by an application such as wget. We performed six Google searches on simple single keywords ("database", "financial", "grades", "homework", "inventory", and "modeling") that are commonly associated with spreadsheets, and requested 1000 results for each of these keywords. To these results we also added 26 spreadsheets previously collected by the Forms/3 group of researchers at Oregon State University, 45 spreadsheets presented in [8] and used to validate methodologies in [1, 2, 3], 13 spreadsheets presented in [11], 9 spreadsheets created by students in an introductory computer science course at Oregon State University and used to validate methodologies in [1], and 5 spreadsheets created and used by the first author. Table 1 (column 2) indicates how many spreadsheets were initially found for each of the sources.

Given these spreadsheets, we needed to determine which ones would be suitable for the kinds of automated experiments that we wish to support. To do this, we performed some initial data collection, measuring various spreadsheet features described in Section 3. During this process we found several spreadsheets that we were unable to use for various reasons. These ranged from being password protected or shared to including disruptive macros that we were unable to disable (often older Excel Workbook 4.0 macros). After gathering information on the various details of the spreadsheets, we then used a two-pass methodology to find and remove duplicate spreadsheets, first utilizing the metrics to group similar spreadsheets together, and then using a Perl script and the Unix utility cmp to find all identical files in the collection, and removing all but one copy of each. Table 1 includes the number of files that we had to remove because they caused problems or were duplicates of other files (columns 3 and 4), as well as how many files remained from each of the sources (column 5).

3 Metrics

Experiments have different goals, and often require spreadsheets with specific characteristics. For example, for basic studies of WYSIWYT, we are interested in spreadsheets that contain formula cells and do not contain macros. However, studies of header inference techniques used for unit inference may not need formula cells. In order to support a wide range of experiments, we collected a wide range of measures on each spreadsheet, as follows:

1. Number of input cells (non-empty cells without formulas).
2. Number of input cells with values of each of the following types: error, boolean, date, non-integer number, integer number, string.
3. Number of input cells referenced by other cells.
4. Number of input cells referenced by other cells with values of each of the following types: error, boolean, date, non-integer number, integer number, string.
5. Number of formula cells.
6. Number of formula cells that evaluate to a value of each of the following types: error, boolean, date, non-integer number, integer number, string, blank.
7. Number of formula cells that contain references to other cells.
8. Number of formula cells that are referenced by other cells.
9. Number of formula cells that use each of the following functions: sumif, countif, choose, hlookup, index, indirect, lookup, match, offset, if.
10. Number of formulas that occur only once in a spreadsheet (according to copy/paste semantics).
11. Number of formulas that occur more than once in a spreadsheet (according to copy/paste semantics).
12. Number of times the most frequently occurring formula occurs in spreadsheet.
13. Whether the spreadsheet includes any charts.
14. Whether the spreadsheet includes any VBA macros.

Tables 2, 3, 4, 5, 6 and 7 give summary statistics for each of the collected metrics.

To collect each of the measures for a spreadsheet s, we implemented an Excel VBA program, `ProcessSheets`. `ProcessSheets` loads s, and iterates through all worksheets in s. For each worksheet w, it iterates through the cells in $w.UsedRange$, an Excel-provided property of worksheets that returns a range of cells guaranteed to include all non-empty cells in the worksheet. `ProcessSheets` then categorizes a cell c as either an input cell (Measure 1) or a formula cell (Measure 5) using the Excel property $c.HasFormula$.

	total	error	boolean	date	non-integer	integer	string
Minimum	0	0	0	0	0	0	0
1st Quartile	141	0	0	0	0	0	70
Median	357	0	0	0	1	47	184
Mean	2967.4	0.4	77.3	70.4	451.6	1313.6	1046.9
Mode	64	0	0	0	0	0	17
3rd Quartile	961	0	0	1	47	215	499
Maximum	3292295	662	169982	51310	724527	2216134	510895
Sum	13314821	1585	347618	316489	2031450	5908512	4689250
Number of non-zero	4486	25	29	1320	2257	3352	4481
Mean of non-zero	2968.1	63.4	11986.8	239.8	900.1	1762.7	1050.9

Table 2: Summary statistics for input cells

	total	error	boolean	date	non-integer	integer	string
Minimum	0	0	0	0	0	0	0
1st Quartile	0	0	0	0	0	0	0
Median	0	0	0	0	0	0	0
Mean	126.6	0.0	0.2	0.5	24.1	90.2	11.5
Mode	0	0	0	0	0	0	0
3rd Quartile	30	0	0	0	0	15	0
Maximum	45911	0	459	1092	18950	40546	12835
Sum	569308	0	1110	2066	108454	405669	51638
Number of non-zero	1609	0	8	66	815	1486	448
Mean of non-zero	353.8	N/A	138.8	31.3	133.1	273.0	115.3

Table 3: Summary statistics for referenced input cells

	total	error	boolean	date	non-integer	integer	string	blank
Minimum	0	0	0	0	0	0	0	0
1st Quartile	0	0	0	0	0	0	0	0
Median	0	0	0	0	0	0	0	0
Mean	167.9	6.1	0.4	0.8	54.9	83.5	15.4	6.9
Mode	0	0	0	0	0	0	0	0
3rd Quartile	50	0	0	0	1	18	0	0
Maximum	26444	3066	1575	362	22315	19353	23104	4891
Sum	744524	27548	1868	3427	246937	375444	69362	30838
Number of non-zero	1977	238	24	136	1139	1730	454	110
Mean of non-zero	382.1	115.7	77.8	25.2	216.8	217.0	152.8	280.3

Table 4: Summary statistics for data types of formula cells

	sumif	countif	choose	hlookup	index	indirect	lookup	match	offset	if
Minimum	0	0	0	0	0	0	0	0	0	0
1st Quartile	0	0	0	0	0	0	0	0	0	0
Median	0	0	0	0	0	0	0	0	0	0
Mean	0.1	0.4	0.0	0.3	0.4	0.2	0.4	0.3	6.3	33.5
Mode	0	0	0	0	0	0	0	0	0	0
3rd Quartile	0	0	0	0	0	0	0	0	0	0
Maximum	180	180	4	450	1016	497	461	331	23104	23104
Sum	450	1786	9	1247	1907	991	1890	1155	28329	150634
Number of non-zero	11	56	3	16	15	4	19	8	8	364
Mean of non-zero	40.9	31.9	3.0	77.9	127.1	247.8	99.5	144.4	3541.1	413.8

Table 5: Summary statistics for functions in formula cells

	referencing	referenced	Size 1 Regions	> Size 1 Regions	Largest Region
Minimum	0	0	0	0	0
1st Quartile	0	0	0	0	0
Median	0	0	0	0	0
Mean	124.6	86.0	6.6	5.2	63.0
Mode	0	0	0	0	0
3rd Quartile	32	11	1	3	16
Maximum	26434	26411	960	414	23104
Sum	560267	386771	29846	23485	N/A
Number of non-zero	1736	1358	1413	1797	N/A
Mean of non-zero	280.3	322.7	21.1	13.1	N/A

Table 6: Summary statistics for formula cells

	Charts	Macros
With	105	126
Without	4393	4372

Table 7: Number of spreadsheets with and without Charts and Macros

To determine whether c is referenced by other cells (Measures 3 and 8, also needed for Measure 4), `ProcessSheets` uses the *c.DirectDependents* property. When c is not referenced by any other cells in the same spreadsheet, trying to access this property causes a VBA error to be thrown, otherwise this property returns the range of cells that reference c. Similarly, if c is a formula cell, `ProcessSheets` uses *c.DirectPrecedents* to determine whether c references any other cells in the spreadsheet (Measure 7). Note that both of these properties are limited to references within the same spreadsheets, so inter-spreadsheet dependencies are ignored.

To determine the type of the value of c (Measures 2, 4, and 6), `ProcessSheets` uses the property *c.Value*, and performs the following series of tests in order on the value until it finds one that is true, and returns the corresponding type:

1. `IsDate`(*c.Value*) – date
2. `VarType`(*c.Value*) = *vbBoolean* – boolean
3. `IsNumeric`(*c.Value*) and *c.Value* = `Int`(*c.Value*) – integer number
4. `IsNumeric`(*c.Value*) – non-integer number
5. `IsError`(*c.Value*) – error
6. *c.Value* = “” – blank
7. otherwise – string

In addition, when c is a formula cell, `ProcessSheets` searches for strings in the formula matching each function, f, using the VBA expression: “*c.Formula* like “*[!A-Z]” & f &“(*”” (Measure 9).

To find duplicate formulas in a spreadsheet (needed for Measures 10, 11, and 12), for each spreadsheet `ProcessSheets` maintains a hash table (actually a VBA collection), using *c.FormulaR1C1* to

index counts of each distinct formula. As indicated by Sajaniemi [16], R1C1 formulas in Excel use absolute-direct, relative-offset referencing and as such do not change when copied to other cells.

To detect VBA macros in s (Measure 14), `ProcessSheets` iterates through each component c in $s.VBProject.VBComponents$, and if $c.CodeModule$ is not Nothing and $c.CodeModule.CountOfDeclarationLines > 0$ for some c, then s is assumed to include macros. To detect charts (Measure 13) in s, `ProcessSheets` looks at $s.Charts.Count$, and if greater than 0, s is assumed to include charts.

We have built a spreadsheet that details all of the measured features for each of the spreadsheets in our corpus, to allow easy selection of subject spreadsheets for studies.

4 Conclusion

We have collected the EUSES Spreadsheet Corpus to allow researchers to validate their methodologies on a standardized collection of "real-world" spreadsheets that has been carefully filtered to allow easy automated processing from within Microsoft Excel. Our hope is that this collection will be useful to researchers for evaluating tools and methodologies for developing spreadsheets. We further expect that as research continues, we will be able to augment this corpus in ways that facilitate types of studies not yet anticipated.

Information about obtaining the EUSES spreadsheet corpus can be found at `http://cse.unl.edu/~mfisher/corpus/`.

Acknowledgements

This work was supported in part by the EUSES Consortium via NSF Grant ITR-0325273. We thank the Forms/3 research group at Oregon State University for providing their collection of spreadsheets for inclusion, Robin Abraham for pointing us to the spreadsheets in [8] and [11] and supplying spreadsheets from the CS101 class at Oregon State University, and Daniel Ballinger for the use of his Java front-end to Google which we used to find spreadsheets on the internet.

References

[1] R. Abraham and M. Erwig. Header and unit inference for spreadsheets through spatial analyses. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, Rome, Italy, September 2004.

[2] Y. Ahmad, T. Antoniu, S. Goldwater, and S. Krishnamurthi. A type system for statically detecting spreadsheet errors. In *Proceedings of the 25th International Conference on Software Engineering*, Portland, OR USA, October 2003.

[3] T. Antoniu, P. Steckler, S. Krishnamurthi, E. Neuwirth, and M. Felleisen. Validating the unit correctness of spreadsheet programs. In *Proceedings of the 26th International Conference on Software Engineering*, Edinburgh, Scotland, UK, May 2004.

[4] Y. Ayalew, M. Clermont, and R. Mittermeir. Detecting errors in spreadsheets. In *Proceedings of the European Spreadsheet Risks Interest Group Annual Conference*, London, UK, July 2000.

[5] Y. Ayalew and R. Mittermeir. Spreadsheet debugging. In *Proceedings of the European Spreadsheet Risks Interest Group Annual Conference*, Dublin, Ireland, July 2003.

[6] M. Burnett, C. Cook, O. Pendse, G. Rothermel, J. Summet, and C. Wallace. End-user software engineering with assertions in the spreadsheet paradigm. In *Proceedings of the 25th International Conference on Software Engineering*, pages 93–103, Portland, OR USA, May 2003.

[7] M. Burnett, A. Sheretov, B. Ren, and G. Rothermel. Testing homogeneous spreadsheet grids with the "What You See Is What You Test" methodology. *IEEE Transactions on Software Engineering*, pages 576–594, June 2002.

[8] G. Filby. *Spreadsheets in Science and Engineering.* Springer, 1995.

[9] M. Fisher II, M. Cao, G. Rothermel, C. Cook, and M. Burnett. Automated test case generation for spreadsheets. In *Proceedings of the 24th International Conference on Software Engineering*, pages 241–251, Orlando, FL USA, May 2002.

[10] M. Fisher II, D. Jin, G. Rothermel, and M. Burnett. Test reuse in the spreadsheet paradigm. In *Proceedings of the International Symposium on Software Reliability Engineering*, pages 257–268, Annapolis, MD USA, November 2002.

[11] M. Jackson and M. Staunton. *Advanced Modeling in Finance Using Excel and VBA.* John Wiley & Sons, Ltd., Chichester, West Sussex, UK, 2001.

[12] J. Reichwein, G. Rothermel, and M. Burnett. Slicing spreadsheets: An integrated methodology for spreadsheet testing and debugging. In *Proceedings of the 2nd Conference on Domain Specific Languages*, pages 25–38, Austin, TX USA, October 1999.

[13] G. Rothermel, M. Burnett, L. Li, C. DuPuis, and A. Sheretov. A methodology for testing spreadsheets. *ACM Transaction on Software Engineering and Maintenance*, pages 110–147, January 2001.

[14] K. Rothermel, C. Cook, M. Burnett, J. Schonfeld, T. Green, and G. Rothermel. WYSIWYT testing in the spreadsheet paradigm: An empirical evaluation. In *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, June 2000.

[15] J. Ruthruff, E. Creswick, M. Burnett, C. Cook, S. Prabhakararao, M. Fisher II, and M. Main. End-user software visualizations for fault localization. In *Proceedings of the ACM Symposium on Software Visualization*, pages 123–132, San Diego, CA USA, June 2003.

[16] J. Sajaniemi. Modeling spreadsheet audit: A rigorous approach to automatic visualization. *Journal of Visual Languages and Computing*, 11(1):49–82, February 2000.