Knowing what was done: uses of a spreadsheet log file

Andy Adler
School of IT & Engineering
University of Ottawa
adler@site.uottawa.ca

John C. Nash
School of Management
University of Ottawa
jcnash@uottawa.ca

February 17, 2004

Abstract

Spreadsheet use in educational environments has become widespread, likely because of the flexibility and ease of use of these tools. However, they have serious shortcomings if the teacher is to understand exactly what students or others have done. It is far too easy for students to replace a formula that gives an apparently unacceptable answer with a number that they believe to be correct. The same concern applies to recorded marks, as well as to business spreadsheets and to other reports that are used for decision-making. While intentionally misleading changes to spreadsheet files receive much attention, simple mistakes are probably more common. Some of these, such as the Trans-Alta Utilities (Globe and Mail, 2003) cut and paste error that cost the firm $24 million (US), have extreme consequences. Few are merely embarrassing. A log file or audit trail, enhanced by suitable filters, can allow both intentional and accidental changes that cause erroneous results to be caught. In order to meet these requirements, we have developed server based software tool ("TellTable") which allows editing, version control, and auditing of spreadsheet files. Users connect to the server using a standard web browser, and are able to access and edit spreadsheet files in a java applet in the browser window. TellTable has been used for a pilot study to maintain marks and course information for a multi-section courses with several instructions and teaching assistants. This paper describes the TellTable software and preliminary results of the pilot test.

Submitted November 2003; revised January 2004; accepted February 2004.

Keywords: collaborative editing, audit trail, version control.

1 Introduction

Spreadsheet files have become increasingly complex software applications in their own right, and may rival large software projects in terms of development time. In comparison to software engineering, which has developed numerous tools and procedures for change management and verification, such as version control and code audits, spreadsheet development lacks many comparable processes. Spreadsheet files, however, being a mix

of data, code, and presentation elements, are perhaps even more difficult to validate. The need for validation of spreadsheet calculations has a concern of many authors. For example, Baker and Sugden [2] address the issue of auditing spreadsheets, and note our report (Nash and Quon [9]; see also Nash, Quon and Gianini, [8]) of deficiencies in statistical computations. Baker and Sugden [2] have reviewed the history of spreadsheets in education. We will therefore focus on issues related to knowing what has been done to a spreadsheet file, that is, to log or audit trail capabilities.
The European Spreadsheet Risks Interest Group (www.eusprig.org) conference proceedings contain many papers on the general subject of auditing spreadsheets. However, most such activities and tools focus on finding erroneous or suspicious features in an existing spreadsheet without access to its history. Alternatively, they may seek to verify correctness of results by building a parallel computation. Recently, Nash, Smith, and Adler [10] described a system to allow a full audit trail of a spreadsheet file to be recorded and analyzed. This system ("TellTable") is installed on a network accessible server which controls user access, manages spreadsheet file versions, and runs the spreadsheet software. Users access the server and manipulate the spreadsheet with a standard internet browser. All spreadsheet changes are archived and can be viewed in their original form, or analyzed with a special viewing program. Currently, TellTable is undergoing its first pilot use. Below we elaborate on the applications of this software in education. We believe TellTable is complementary to other spreadsheet audit tools.

2 TellTable implementation

TellTable is a consolidation of a number of existing open-source software elements "glued" together with custom software and augmented with application specific tools to enable efficient analysis and audit. We anticipate that while some of the analysis tools will be open-source, others will be offered for sale or licence as proprietary software. We intend the base system to be released under the Gnu Lesser General Public Licence (Free Software Foundation [6]). This system is available on the SourceForge repository under the name "telltable-s" (accessible at telltable-s.sourceforge.net).

Spreadsheet processing in TellTable is carried out with OpenOffice.org ***calc***. We did not have to modify the source code of this product, which is available for free download from openoffice.org. However, in order to control against users disabling the audit trail, we modify parts of the XML files that allow the customization of the display to remove the menu items that allow unwanted features.

Using the Java VNC applet (Virtual Network Computing, Richardson *et al.*, [12]), we are able to present users with the "standard" ***calc*** interface within a browser window. Thus users can be working on various platforms. Indeed, the authors of TellTable work on various Windows, Unix/Linux and Macintosh platforms.

Our prototype server was a Pentium II 266 Mhz PC running Mandrake Linux version 9.1. For security, the TellTable system creates internal user accounts that do not have access to operating system functions. Each system user is assigned to an internal user account when working with the ***calc*** software. Versioning is accomplished using the well-

known and widely-used CVS system (Cederqvist [5]), for which we provide a web-based interface integrated with TellTable functions and options.

The audit trail for each file is kept within that file via the change recording function of the ***calc*** software, so that downloading a particular copy of the file includes information about all changes to the file. In order to view and analyze this change record data, we have developed a viewing and analysis tool. This is currently over 10000 lines of Java code and offers many features to filter change records by date, user, and type of change. For example, we can look for all changes where a formula has been replaced with a static value.

Our reason for designing this system to operate on a server is that we can think of no method that reliably controls against hiding of changes if the spreadsheet is processed locally. The **TellTable** interface does, however, allow for download of files. The software offers the option to "edit" or "download" each file that a user may access. Downloaded files may be processed locally, but we do not yet have a system for re-integrating such files into the server-based collection. Clearly, there are a number of philosophical issues in permitting such re-integration, but we recognize that this is a feature that will be needed in some form.

At the present time we operate TellTable only on a Linux server platform. There is no particular reason that other platforms (Unix, Windows, Macintosh) could not be used, though appropriate versions of the software tools (***calc*****, CVS, VNC,** some database and utility functions, as well as our scripts, written mainly in Perl) would be needed. In particular, a port to Windows would require significant modifications, due to the very different security model for the graphical user interface. We have no current plans to carry out such a port, but would welcome discussions with those interested in so doing. Since the client may be on any platform, and we recommend that TellTable run on a server that carries out no other functions (to avoid security risks), a port is not urgent. One business model for TellTable is that we would supply an office suite "appliance", much like a network router, since we have been using other tools within the OpenOffice suite on the same server (Adler, Nash and Noël [1]).

Our choice of ***calc*** was driven by the availability of source code and documentation for the software, so that we could make the necessary adjustments to ensure the audit trail is kept. To simplify our work, we have so far only edited and saved the native ***calc*** ".sxc" file format, which stores its data in a compressed XML format. However, ***calc*** does read and save Microsoft Excel format spreadsheets, though we believe some experimentation is needed before we can be sure that audit trail information is not lost in the conversions. We also note that the ".sxc" files are considerably more compact than their Excel ".xls" counterparts. Our experience is that our users think the spreadsheets are "Excel" files, and only note the difference if they download them.

Our current versioning uses CVS. As this is more or less a "plug-in" element via the Perl scripts that present the TellTable user interface, it should not be difficult to replace it with other version-control software. We have no plans to do this at the present time, even though CVS evolved as a tool for control of plain-text program files, and the compressed spreadsheet and other OpenOffice files are binary. This means that the "diff"

functions of CVS (which calculate the differences between versions of text files) do not work on the binary spreadsheet files. Such capability would be of great benefit to users, but is significantly more complex than for text files. Document differences would need to be determined at the application level. Our audit software already provides a certain amount of this capability.

3 TellTable Applications in Education

3.1 Mark recording and archiving

A common current practice is to manage marks by emailing spreadsheet files between the various teaching assistants (TAs). The principal problem of this practice is the version management difficulties: independent changes can be made to different versions, which must later be reconciled manually. It is also difficult to determine when a change was made, and why. Additionally, serious errors can be made very easily, such as the pasting of an entire list of marks over the wrong rows. The latter error has been encountered more than once in the past, with both of us the unwitting perpetrators.

This application has been our first major test of TellTable. Our users are the teaching assistants who must enter marks. TellTable allows several TAs to access the same file, though not at the same time, with a record of which TA altered which cells, the change in the content, and the timestamp of the change. Thus we have a complete record of changes. Moreover, TellTable includes a complete versioning system, so that the "manager" or senior professor may revert to any saved version of the marks file(s) if there is a serious error.

Mark recording is clearly a useful application of the TellTable logs, especially if we filter the output to focus on changes which result in outcomes especially beneficial or detrimental to a few students. In particular, the replacement of formulas with static values will be of interest. Not all such changes are the result of dishonesty – there are genuine cases where the professor will need to make an *ad hoc* correction to a mark.

The preliminary result of our usage of TellTable for mark recording has been that the TAs found it relatively easy to use, even over a dial-up line. Since TellTable runs OpenOffice.org ***calc*** on a server and communicates the user screen via a web-browser using VNC (Virtual Network Computing) technology, the satisfactory operation over dial-up lines was a pleasant surprise. The main complaint, which we are addressing, is that we made the server "too secure". First, if one TA has a file in use, others are locked out. Should the first user not log out properly, the file remains locked. We are addressing this issue with an automatic logout on non-activity or other events that indicate the session is over. Second, there were complaints of failure that were the result of bugs in our code. At the time of writing both issues have been corrected and we are in the second phase of our test.

If there is a weaknesses in the technology that our test has revealed, it is that the system does not properly allow cut and paste from existing files local to the user's machine into the server based spreadsheet. This problem is inherent to use of a Java applet in

a internet browser based client. The security model of the Java virtual machine does not allow cut and paste interaction with the local machine. thus, this is a problem that is presented by the use of a web-browser interface, and is more general than our own software. However, it has particular implications for setting up spreadsheet files for use, for example, in providing the base classlist files for TAs to use for marks, or for resetting a file should an older version be needed. There are several possible approaches to solve this issue, such as using a technology like ActiveX to embed the applet into the internet browser, which does not have the restrictions of the Java applet.

3.2 Teaching spreadsheet use

Many educational institutions teach students how to use spreadsheets. Without an audit trail, however, there is no way to learn how the student built the spreadsheet they submit to the teacher for analysis or evaluation. It is well-known that even experienced users make errors in constructing spreadsheets. For example, Panko [11] and Irons [7] asked participants to create a spreadsheet to estimate the cost of a construction job ("The Wall" and "The Wall and Ball", respectively), and observed a surprisingly high rate of errors (average cell error rates for "The Wall" were 1.67%, while for the "Wall and Ball" problem a shocking 11.86%). Even worse, the participants in this research were typically unaware of their errors (Panko [11]). While effectively teaching spreadsheet use to correct these errors is difficult, we believe that the availability of an audit trail would provide an important pedagogical tool, making it possible to determine the ordering of steps and the manner of error-making in which students or others construct their spreadsheets.
So far we have not attempted such an analysis, since we do not ourselves instruct spreadsheet use. However, we sincerely hope to engage with others in such activities, and welcome collaboration in this regard. Moreover, since TellTable is built on the OpenOffice.org ***calc*** files, which are zip-compressed XML text, it is reasonable to envisage extending or modifying our current TellTable-View scanner program to aid in this activity. We note that others are already looking at automatic evaluation of student programming efforts (Bull *et al.* [4]), but their efforts focus on the completed student submission rather than explore the route by which such files were constructed.

3.3 Spreadsheets in modelling for decisions

In commerce and industry, spreadsheets are frequently used for modelling costs and revenues as a basis for decision-making. Sometimes errors are made, as in the recent Trans-Alta Utilities case (Brethour [3]), which cost that company $US 24 million. Educational activities mirror the outside world, and spreadsheet models are used in various management courses for exercises that simulate the real-world activity. We built TellTable with the intent of using it to be able to examine how students carried out their "modelling". While a great deal of detailed analysis is possible, there is also the simple usage in verifying that a student has not simply over-written a formula calculation they believe to be unacceptable with a static number "borrowed" from another student. The

audit trail should also show a compressed time-line if students have used a block cut-and-paste from another student's file. Clearly the former of these situations is academic fraud, so our use of TellTable in this way is defensive. The latter may be plagiarism in many circumstances, depending on the requirements of the assignment. Hopefully, once students know we can detect such fraud, we can move on to more positive applications that will improve teaching and learning.

4 Use of the TellTable tools

This section illustrates a sample session using the TellTable tools discussed here. Data and images are taken from the current pilot project, in which spreadsheet files are being managed for the courses taught by the authors at the University of Ottawa. For illustration purposes, we consider the courses taught by Adler: Electronics III and Electronique III (the same course material but in English and French). These courses require five assignments, a midterm and final exam and an electronics project. The project requires three "milestone" presentations, at which different parts of the electronics function need to be demonstrated, as well as a final report. The courses have a combined 120 students, and four teaching assistants (two full time, and two part time) are allocated (only one of whom can speak French). Thus, in order to manage the course, the following documents need to be managed:

1. the course marks for each activity,
2. the list of project groups and the presentation timetable for each "milestone", and
3. the list of which TA is taking care of which activity.

Each of these documents is dynamic: For example, TA tasks may need to be shuffled if their availability changes, or new tasks are required; project groups change as some students find they are no longer able to work together; and course marks as new work is graded, and also if students identify errors in returned work. In all of these cases, it is important that the instructor and TAs always have access to the most recent version of any document, and any changes to the documents can be identified to the authors.
In order to access the documents, the instructor or teaching assistant is assigned a username and password. They then type the appropriate URL into a web browser, and are presented with a log-in screen (not shown). Within the system, each user is assigned membership to various groups, which determine which files may be viewed and edited, and whether the user has access to the version history of documents. Figure 1 shows a user "testuser" with permission to access the electronics course marks. After logging into the system, "testuser" is presented with the screen shown in Figure 1. This screen presents the files available to the user: “course-marks”, “project-test-list”, and “TA-joblist”. The latest modification date for each file is shown. Clicking on the link next to "Edit" will lock the file and send the user to the spreadsheet editing screen (Figure 2). Note that the file "project-test-list" is locked and not available for editing. All files

are available for Download. This is a read-only function that allows the user to view or print the contents.

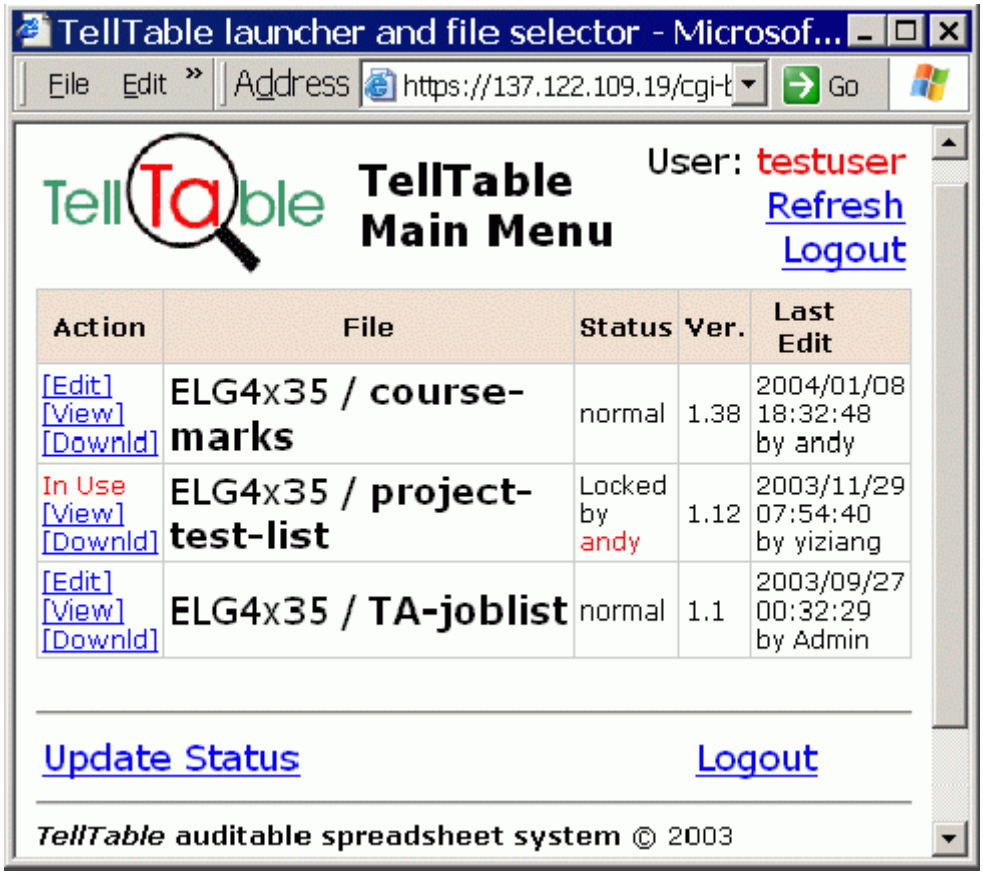

Figure 1: TellTable document selection screen. Users are presented the option to edit, view or download spreadsheet files. The file "project-test-list" is currently in use, and is locked against editing by another user.

After selecting "Edit" on "course-marks", the user is taken to the spreadsheet editing screen shown in Figure 2. This window contains some dynamic text content at the top, with a link to return to the application after the document is saved. In the center of the window is a Java applet containing a view of the OpenOffice.org ***calc*** spreadsheet running on the server; all mouse and cursor activity is sent to the server to be processed, and the results returned to this screen. This means that this application allows full spreadsheet functionality and flexibility. Users can add data, formulae, graphs, and other sophisticated spreadsheet elements. After finishing editing the document, the user will save it, exit the spreadsheet program, then exit the current web page to return to the page shown in Figure 1. We are working to remove some of these extra steps that have been required so far for security of the data. The server will take the document

and store it in the CVS version repository. At this point the user may edit another document, or quit the application. A user with sufficient system privilege is permitted access to the administrative functions of the TellTable application. One of these is access to the version control repository, managed by CVS (Cederqvist [5]). After the user finishes editing a document, CVS tests whether it has been modified, and if so, updates the repository with the latest version, while retaining the information about all previous document versions. Figure 3 shows a screen presenting an interface to the CVS repository, showing the status of the documents, and allowing the change history to be browsed, and previous versions downloaded. We currently do not allow editing of previous versions; we anticipate that only read-only access will be typically needed to answer most application requirements. However, we plan to provide an interface to the "restore" function of CVS (which will restore a file to its contents during a pervious revision) to return to a previous version of the file should a serious situation arise.

Interestingly, CVS does allow the possibility of multiple user checkout of documents, although we do not use that capability here, but rather restrict documents to be locked to a single user. This is because in order to support multiple user checkout of documents, there needs to be a clear application level understanding of how to identify, merge and manage conflicts between these versions. This is a challenge that we feel is not sufficiently well understood to implement properly at this stage. The administrative functions of the application allow adding new users, querying system status, changing group memberships, adding new files, and backing up the version control database. At this point, some of these functions still need to be performed using the command line interface at the server, but will shortly be added to the web based application functions.

Once a spreadsheet has been processed using the secure server, we can examine its change history. Clearly, looking at every cell alteration is a tedious and generally unrewarding task, so our colleague Neil Smith has developed ***TellTable-VIEW*** which is designed to assist in examining the change information. This program allows, in particular, for ***filters*** to be applied to limit the displayed transactions to those of a particular type, for example, formulas which have been replaced with static values, to those made (or not made) by specified users, or those made within or outside a given time period. Figure 4 shows a typical output screen after filtering. Here we have used a constructed example, as our marks example has so far had a limited number of transactions of interest to an audit application. A large number of changes (197) have been made to the original spreadsheet file. In Figure 4, filters have been selected to exclude cell transitions from empty to any value. This filter setting will remove the most common, and innocuous change ? filling in empty cells. Change records that pass the filter selection criterion are then presented to the user. At this point, the number of records presented is sufficiently small, that it is appropriate for manual auditing. We anticipate that, in general, more specific filter settings would be used, in order to search for application specific kinds of suspicious activity. For example, cells can be selected which were modified from formulae to numeric values. TellTable-VIEW currently comprises about 10,000 lines of Java, and makes use of the Xerxes XML parser to extract information from the OpenOffice.org ***calc*** spreadsheet. It does not use ***calc***

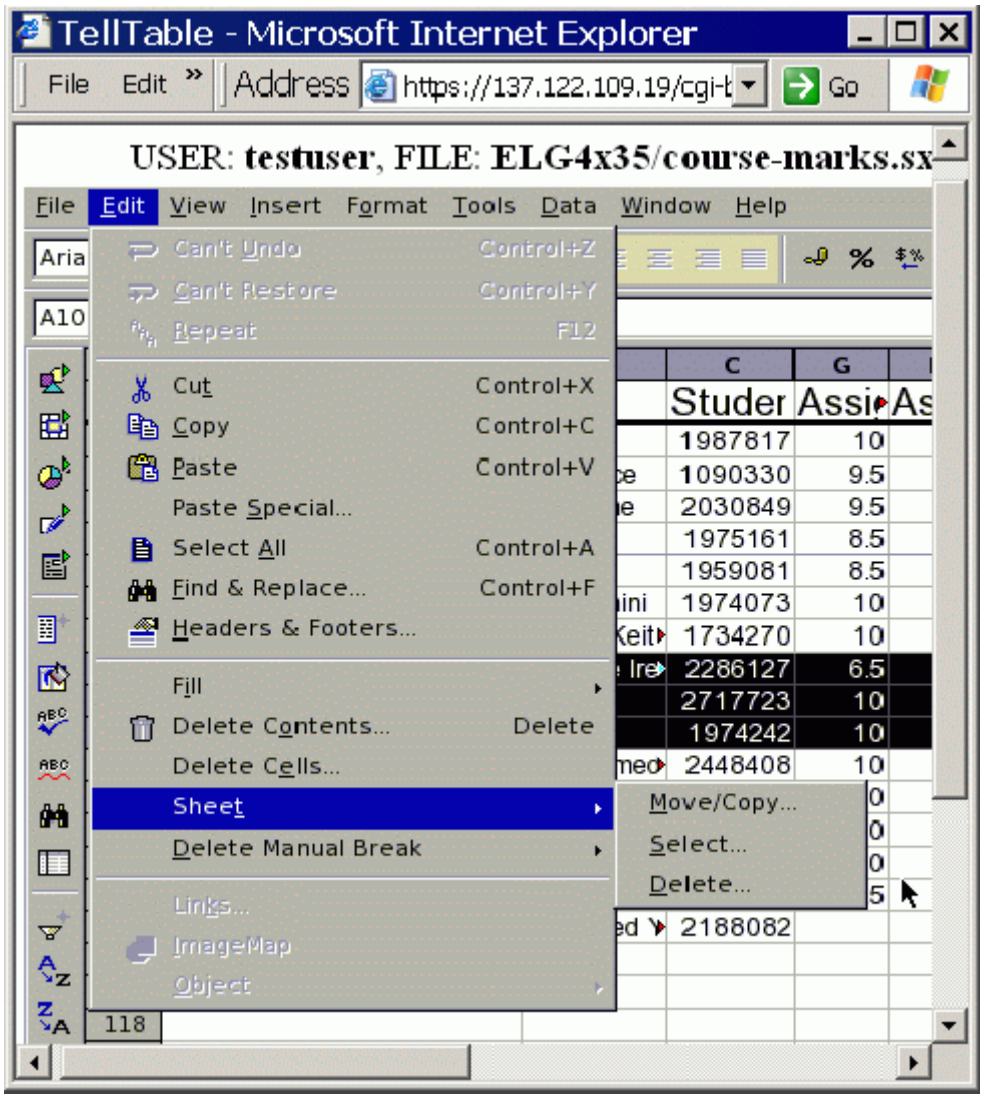

Figure 2: TellTable spreadsheet editing screen. The spreadsheet software is running on the web server, and the display is being exported into a Java applet in the web browser window. Full spreadsheet functionality is available, but with some security restrictions.

Figure 3: TellTable ? version control interface. This screen is available to administrative users, who are able to view previous versions of files, as well as when they were edited and by whom.

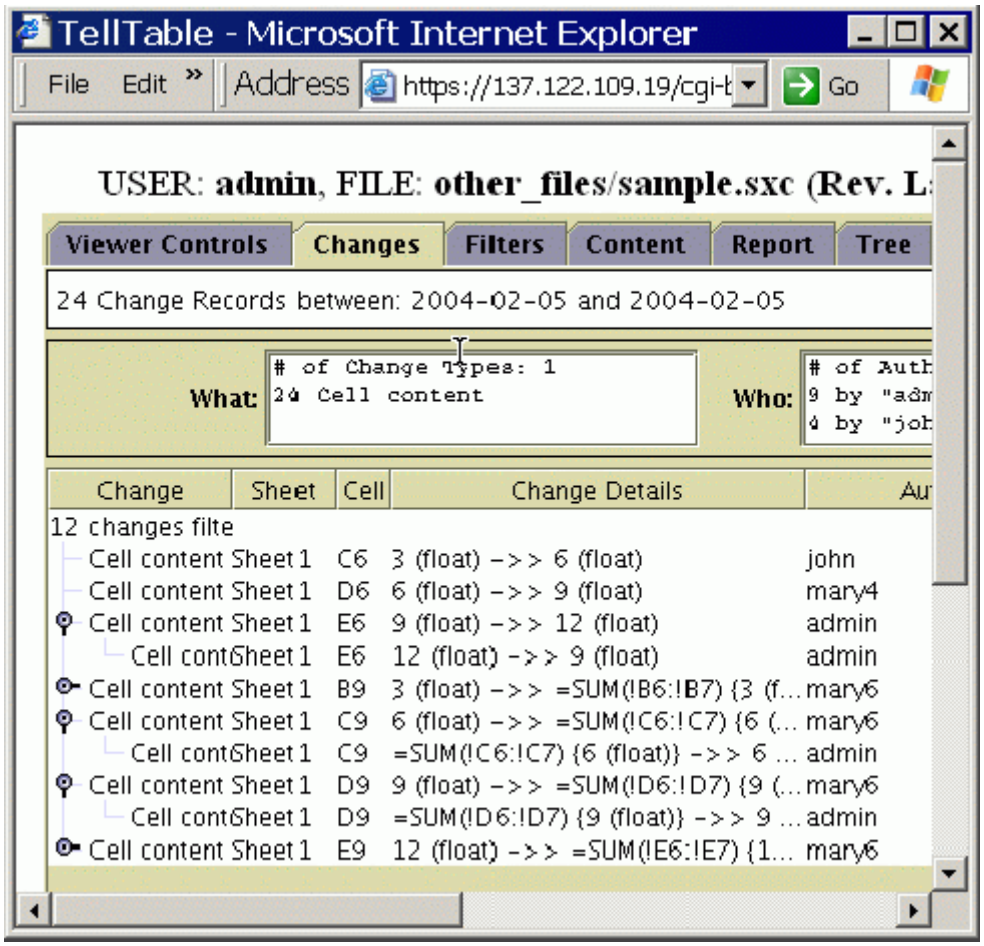

Figure 4: TellTable ? VIEW example screen shot. Each modification to the spreadsheet file is processed and compared against a set of filters (detailed under the "Filters" tab). Each line which passes the filter criterion is shown in the "Changes" set, as shown. Thus, of 197 spreadsheet changes, this screen shows that a combination of filters on the author and content has removed the majority of change entries.

to read the spreadsheet file, and is designed to only read such files, since we believe the audit function should be carefully isolated from editing.

5 Conclusion

We have developed a software application that allows change control and audit of spreadsheet files. This software has been build using a large number of pre-existing components, which, together, provide a novel and useful capability. We feel that this approach allows two opposite dangers to be avoided: 1) inflexibility, as is seen in many "enterprise" applications, which are often left unused in favour of "black book" spreadsheet files, and 2) "the version management nightmare" of emailed spreadsheet files. TellTable provides all the flexibility of spreadsheet software, while maintaining access and version control, detailed audit logs, and validation. In other words, greater auditability permits flexibility while maintaining security.

We are acutely aware that our capability to provide a spreadsheet audit trail is so new that there has not been sufficient time to apply it widely. We have, however, established the web site www.telltable.com where descriptions of ongoing work and links to papers and presentations are given, and we have registered the project “telltable-s” at the sourceforge.net repository, where the open source components of this software will be made available for download or collaborative development. In particular, visitors will note that TellTable is advancing with the cooperation of workers with academic, private and government affiliations.

References

[1] Adler, A. Nash, J. C. and Noël, S. (2004) Issues in collaborative document creation, in preparation.

[2] Baker, J.E., and Sugden, S.J. (2003). Spreadsheets in Education: The First 25 Years. *Spreadsheets in Education* **1**(1): 18–43.

[3] Brethour, P. (2003). Human error costs TransAlta $24-million on contract bids. *Globe and Mail*, Toronto, Wednesday, Jun. 4, 2003 (internet edition from http://www.bpm.ca/TransAlta.htm).

[4] Bull, S., Greer, J., and McCalla, G. (c. 2003) The caring personal agent. I*nternational Journal of Artificial Intelligence in Education*, accepted for publication. See http://www.cs.usask.ca/research_groups/aries/publications.htm

[5] Cederqvist, Per (2002). Version management with CVS, Bristol UK: Network Theory.

[6] Free Software Foundation (1999). GNU Lesser General Public License version 2.1, http://www.gnu.org/copyleft/lesser.html.

[7] Irons, Richard (2003). The wall and the ball. *Proceedings of the 2003 EUSPRIG Conference* (European Spreadsheet Interest Group), David Chadwick and David Ward, editors, 33–48.

[8] Nash J.C., Quon, A., and Gianini, J. (1995). Statistical issues in spreadsheet software, *1994 Proceedings of the Section on Statistical Education*, American Statistical Association, Alexandria VA, 238–241.

[9] Nash, J.C. and Quon, A. (1996). Issues in teaching statistical thinking with spreadsheets. *Journal of Statistics Education* **4**:1, March 1996.

[10] Nash, J.C., Smith, N., and Adler, A. (2003). Audit and change analysis of spreadsheets. *Proceedings of the 2003 EUSPRIG Conference* (European Spreadsheet Interest Group), David Chadwick and David Ward, editors81–88. Also School of Management, University of Ottawa, Working Paper 03-23.

[11] Panko, R. (2003). Reducing overconfidence in spreadsheet development. *Proceedings of the 2003 EUSPRIG Conference* (European Spreadsheet Interest Group), David Chadwick and David Ward, editors, 49–66.

[12] Richardson, T., Stafford-Fraser, Q., Wood, K.R, and Hopper, A. (1998). Virtual Network Computing. *IEEE Internet Computing*, **2**:1, 33–38. See also www.uk.research.att.com/vnc/, www.realvnc.com/vnc/, www.tightvnc.com/vnc/.