SheetGit: A Tool for Collaborative Spreadsheet Development

Ricardo Moreira

NOVA LINCS, DI, FCT, Universidade NOVA de Lisboa, Portugal
rm.moreira@campus.fct.unl.pt

Abstract. Spreadsheets play a pivotal role in many organizations. They serve to store/manipulate data, and are often used to help in the decision process of organizations, thus having a direct impact on their success. As the research community already realized, spreadsheets tend to have the same issues "professional" software has.
One of the most used mechanisms to manage software projects is version control. Thus, we present a version control system oriented for end-user programmers. It allows for seamless and risk-free collaboration between users, to graphically visualize the history of spreadsheet versions, to switch between different versions just by pointing and clicking, and to visualize the differences between two spreadsheets in an animated way.

Keywords: Spreadsheets, End-user Software Development, Microsoft Excel, Version Control, Excel Add-in

1 Introduction

There have been multiple studies attempting to measure errors in spreadsheets, and they have always found them in abundance [8]. While spreadsheets are easy to share and modify, this actually makes it difficult to control and maintain their integrity [4]. The amount of user controls in spreadsheets do not approach the level of controls that professional programmers have found to be necessary for their development, so errors are more likely to happen and not be detected. Indeed it is common for spreadsheets to reach high levels of complexity and size, making them hard to comprehend and debug, ergo increasing the number of errors when they are used [2].

Essentially, spreadsheets are being used as cheaper, more agile replacements of professional programs that would normally cost large sums of money to create and maintain, but they have fewer controls and tools to prevent errors. If professional programmers are no longer making sure end users will not make mistakes, to prevent these from happening, end users must themselves start to adopt the disciplines and tools that professional programmers have long used when dealing with complex software [9]. One of these mechanisms is *version control*.

Version control is known to be extremely beneficial for experienced programmers [7], but it is also beneficial for end-user programmers, both for learning

and debugging purposes [6]. Version control also helps in understanding spreadsheets as these reach high levels of complexity. With proper version control, one can see how the spreadsheet was built over time and, from its origin, gradually come to understand it. Moreover, this is the way professional programmers collaboratively work, and thus, it should also be used by end-user programmers.

Our goal is to bring version control to spreadsheet end-user developers, in an intuitive manner, to help modernize Excel's development tools, to help lower the risk of spreadsheet errors, to help end users create, manage, comprehend complex spreadsheets and to enable easy, risk-free collaboration and sharing of spreadsheets. For this purpose, we have created an add-in for Excel which automatically creates versions, branches, allows users to change between versions, and see the differences between them. Since end-user developers are the target demographic, the user interface is the prime focus of the application, and we intend to have it thoroughly validated to ensure it is easy to use.

2 SheetGit

In this section we introduce the tool termed SheetGit we have implemented as a version control solution for spreadsheet end-users. In this tool we have implemented a set of fundamental features: creating versions, switching between them, the option to see the differences between two versions, and sharing them through the Internet. In the next sections we desccribe each of these.

2.1 Showing Versions

Creating a proper user interface is challenging because the project's target audience consists of end users who most likely have never interacted with any type of version control. As a result, we designed the interface to be simple and intuitive, and for most of the features to be automated or easy to understand with a fast tutorial, though it still requires validation through an empirical study.

We chose to use a tree to display our versions, as seen in Figure 1, because of its simplicity and adaptability, allowing for an intuitive graphical representation of many advanced version control functions such as branches and merges. Very popular version control services such as Github [5] or Bitbucket [1] also make use of a tree structure to display their version lists.

The versions are placed chronologically, the large white version being currently active, meaning it is the spreadsheet the user is seeing. In this case, the user restored his spreadsheet to a previous point in time, so it is not the most recent version of the spreadsheet.

2.2 Creating Versions

The default behavior for creating a version is to do so automatically, as to prevent any sort of data loss due to the user forgetting to save, and to not interrupt the user as he/she works. Since the number of versions may inflate greatly due to the

Fig. 1. A single branch tree displaying a set of commits

automated behavior, we have also implemented a method to combine or prune them where we believe the end result may be indifferent to the user, using metrics such as time since last edit and amount of data entered into the workbook. This only collapses those versions, but the user can always re-open them to a more detail view.

Branches are created automatically when users edit a version that is not at the last one. This will allow them to keep working seamlessly yet still providing a clear view of their current work parentage.

By default, all versions will have a summary of its changes stored in an additional text file, and also displayed on the version tree, as seen in Figure 2. Users can then understand what actions occurred without having to open the version or write a version message beforehand. Nevertheless, the user has always the possibility of adding a version message.

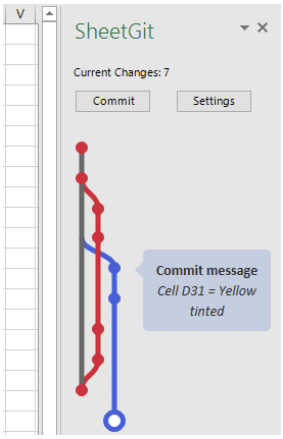

Fig. 2. Our mock up's version tree displaying a commit's details

2.3 Switching Between Versions

Another feature SheetGit supports is to switch from one version to another. To do so, the user has only to click in the desired version in the tree showing the versions. This will change the workbook so it is in the state of the particular version selected.

2.4 Showing Differences Between Versions

Showing differences between versions is one of the greater challenges of the project because, given the free-form factor of spreadsheets, the user is able to organize information in whatever way he feels best. Also because its authors may have never seen *diffs* in a version control system, so various concepts that seem normal to professional programmers can be difficult for some end users to understand.

Differences between versions are shown inside Excel itself. If the user selects a particular version, the mouse/cursor will recreate the actions required to turn one version into the other. If a user could not understand the list of changes in a version, he/she can visualize the animation to see all of them being performed one by one. There are options to speed up or skip the animations to not hinder the more experienced users. As a better illustration of this concept, we refer the reader to the tool's website, at `http://spreadsheetsunl.github.io/sheetgit/`, where a concept movie can be seen showing this mechanism.

Although the tool stores the changes performed between two consecutive versions, for versions more distant, we intend to use *SheetDiff* as our algorithm for detecting differences between versions as it has been proven to be very robust and efficient at discerning changes [3]. This algorithm will find the differences between two versions and we will display them as the set of changes necessary to switch from one version to another.

2.5 Collaborative Development

Users will have a shared repository with other people. To minimize possible conflicts when more than one person is developing the same spreadsheet, each user will be forced to use his/her own personal branches, created from the main branch or from his/her own other branches. When the user is satisfied with the changes, he/she can merge to the main branch such changes so they become incorporated in the main development and available to other users. When merging conflicts may arise. If there is a conflict in a particular cell, such cell can show the conflicting values and the corresponding version, through a list, allowing the user to select the intended choice. If an entire row or column is in conflict, then both can be shown and the user is guided to delete the wrong one.

Every version and branch is tinted with a unique color belonging to a specific user, with the master branch always being gray to show it can be used by multiple users, as seen back in Figure 2.

This online portion of the application is optional but is fully automatic once the user grants SheetGit the appropriate permissions to their Bitbucket account.

2.6 Technological Choices

We have have created our solution as an Excel add-in to keep it as closely knit to Excel as possible, as this potentially increases its acceptance among spreadsheet developers as opposed to being run as an external tool. This will also enable us to present information directly in the active spreadsheet, and grant us access to Excel's proprietary file formats.

The add-in functions as a self-contained task pane, that includes an embedded browser to allow us to make use of Javascript, while retaining the advantages of the more powerful C# add-ins. Our version tree is created using Gitgraph.js[1], a Javascript library.

The collaboration side of the application will be performed with the help of Bitbucket, using it as the online hosting service for the spreadsheets. This because Bitbucket offers free registration, a welcoming API, and private repositories, allowing users to maintain their privacy.

Availability SheetGit is available as a Microsoft Excel open-source add-in at `http://spreadsheetsunl.github.io/sheetgit/`. Since this is an open-source project, we believe it will receive contributions from the community, and thus will be constantly improved.

3 Related Work

Many version control tools already exist to work with spreadsheets, such as Microsoft Sharepoint's *History*[2], Google Sheets[3], XLTools's Version Control[4] and Pathio[5], and while they all perform well, they generally have issues when it comes to perceiving changes in formatting and formulae.

They show their versions in the way of a chronological list, but this makes it difficult for users to perceive the parent of a version, which as been shown to be beneficial for end users [6]. Users tend to solve problems by searching for alternatives and then backtracking when required, which is easier when the parent is accessible [6]. With our tree and branch presentation, it is easy for users to keep track of the origin of all changes.

Also when it comes to presenting the differences, it is either done in a very simplistic manner, or in a more complete manner, but end users without version control experience might have a greater difficulty in understanding the presentation without a very rich tutorial.

In regards to collaboration, some tools exist with varying degrees of completeness in its implementation. Google Sheets, after each change, a new revision is automatically created. When viewing the version history, each user is assigned a

[1] `http://github.com/nicoespeon/gitgraph.js`
[2] `https://products.office.com/en-us/sharepoint/`
[3] `https://www.google.com/sheets/`
[4] `https://xltools.net/`
[5] `https://www.pathio.com/`

unique color, and the altered cells are highlighted in the color of its author. This means, however, that what exactly changed is not explicitly shown. Moreover, in Google Sheets everything every user does is visible to every other user. Our apporach allows to have private branches and merge only what is desiered. Microsoft Sharepoint's revisions are created automatically just like Google Sheets but there is no method to view the differences between versions.

4 Conclusion

A version control system is a very helpful tool for professional software development as it allows for both collaboration and controlled development. Such features are also desired when developing spreadsheets by end-user developers. It is however necessary to adapt such systems to the necessities and restrictions of this kind of developers.

Although more features can be added, we believe that the next important step is to validate our proposal with spreadsheet end-user developers. Such an evaluation is complex and should be done in a long term study where the potential benefits would be assessed in the usage of the tool in a considerable period of time ideally in a collaborative development environment.

Acknowledgements This work has been partially supported by NOVA LINCS through the FCT project with reference UID/CEC/04516/2013.

References

1. Atlassian: Bitbucket. `https://bitbucket.org/`, accessed: 2016-03-03
2. Bradley, L., McDaid, K.: Using bayesian statistical methods to determine the level of error in large spreadsheets. In: Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on. pp. 351–354. IEEE (2009)
3. Chambers, C., Erwig, M., Luckey, M.: Sheetdiff: A tool for identifying changes in spreadsheets. In: Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on. pp. 85–92 (Sept 2010)
4. Deloitte: Spreadsheet Management: Not what you figured. `http://www2.deloitte.com/us/en/pages/audit/articles/spreadsheet-management.html` (2009), accessed: 2016-01-18
5. GitHub, Inc.: Github. `https://github.com/`, accessed: 2016-03-03
6. Kuttal, S.K., Sarma, A., Rothermel, G.: On the benefits of providing versioning support for end users: An empirical study. ACM Trans. Comput.-Hum. Interact. 21(2), 9:1–9:43 (Feb 2014), `http://doi.acm.org/10.1145/2560016`
7. L.Mitchell: You're not using source control? read this! `http://www.lornajane.net/wp-content/uploads/2013/01/source-control-whitepaper-v1.1.pdf` (2014), accessed: 2016-01-11
8. Panko, R.R.: What we know about spreadsheet errors. Journal of Organizational and End User Computing (JOEUC) 10(2), 15–21 (1998)
9. Panko, R., Halverson, R.P., J.: Spreadsheets on trial: a survey of research on spreadsheet risks. Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences 2, 326–335 (1996)