Using Overleaf for collaborative projects: First impressions and lessons learned

Boris Veytsman

1 Introduction

The COVID19 pandemic has changed many things. Among them are the ways scientific papers are written. In the past, an author could assume to have physical meetings with the coauthors, to exchange written pages, ideas and thoughts. Unfortunately today these meetings also may mean exchanging virion particles. Thus many collaborations now rely exclusively on virtual meetings. Anybody who has participated in such meetings knows they cannot compare with the lively back and forth of in-person interaction. This puts a heavy burden on the technology involved: we need to compensate for the deficiency of the virtual collaboration with our tools.

In the first months of the pandemic I participated in two completely virtual collaborations, resulting in the papers [1, 2]. One paper has seven coauthors, the other has four. The proper organization of the writing process was therefore very important for us.

Collaboration tools should have several important features. At a minimum they should track the changes in the manuscript and the contributions by the different authors. They should be able to typeset the manuscript at each stage of the preparation. Ideally they ought to provide a way for the coauthors to exchange metacomments about the text.

In the past my tool of choice was the combination of a local TEX installation (TEX Live) and a version control system (*git*, *svn*, *cvs*). This is a very good solution for typesetting and change tracking. On the other hand, it is not a good solution for metacomments, unless one was willing to put the paper on GitHub and use their issue-tracking system. Frequent physical meetings between the coauthors somewhat alleviated the lack of metacomments: the coauthors could always give their comments in person.

The big disadvantage of this workflow is the rather demanding requirements on the coauthors. They need to be not only comfortable with TEX (this is something to be expected from math and science people); they also need to have and maintain a local TEX installation *and* be familiar with a version control system. Personally, I think the ability to use version control is an essential skill for anybody working professionally with texts. Unfortunately, the reality is that this knowledge is rare outside the programming community (and even in this community it is often rudimentary; see Figure 1).

Figure 1: *Git*, from `https://xkcd.com/1597/`

Overleaf (`overleaf.com`) was suggested as a collaboration tool. We used it for the papers mentioned above. In this paper I discuss our experience and the lessons learned.

2 Overleaf workflows

There are several ways of working with Overleaf. First, one can use its native interface; each coauthor logs into the site, uses its online editor to create TEX files, uploads the images for the figures, and uses the cloud TEX installation for compilation. Some of my collaborators chose this workflow. However, it was not convenient for me. As a person who has spent countless hours with my trusty Emacs, I cannot imagine writing texts with anything else (an obligatory aside: I have complete respect for my *vi*-using friends). Also, I did not trust the Overleaf version control system, so I wanted to use my own.

Fortunately, Overleaf provides another workflow (currently only for paid accounts — but see below). Namely, it can serve as a *git* remote server. You can do a `git clone` for an Overleaf repository, and then use the familiar `git push` & `git pull` commands to sync the local *git* repository with the one at Overleaf. This workflow has the additional advantage of being very flexible; some coauthors can use the native Overleaf interface, while some coauthors can use *git*, and the two are smoothly integrated.

It should be noted that Overleaf offers yet another workflow: integration with GitHub rather than a local user's repository. However, we considered this scheme too complex, and we did not use it.

3 Overleaf access control

Whether the resulting paper is freely distributed or not, most authors would not like to show the world their unfinished work, with all its embarrassing errors, vague ideas postponed for the future papers, heedless statements, etc. Thus it is crucial for a collaboration tool to support defining who has the right to read and edit the manuscript.

Overleaf has two modes of access control. The first mode, available for both free and paid accounts, is based on link sharing. A project has a link for read/write access and a link for read-only access. Anybody in possession of these links (each being a long combination of random letters and digits) has the corresponding rights.

In the second mode, one can give the rights (read/write or read-only) to the chosen Overleaf users (who must log in to Overleaf to participate). The maximum number of collaborators for each project depends on the account tier: from one collaborator for free accounts to unlimited collaborators for "Professional" ones. It is important to note that the limit is determined by the tier of the project owner: if the owner has a "Professional" account, any number of other collaborators can have free accounts. The same is true for the *git* integration discussed above: if the project owner has a paid account, all other collaborators can use *git*.

In general, the Overleaf access control model seems to be mature and corresponds to the current industry best practices. The possibility to use third parties' credentials to login to Overleaf (IEEE, Google, Twitter, and ORCID) is also a convenient feature.

4 Overleaf TEX

To tell the truth, I had my doubts about the Overleaf TEX installation. My TEX is sometimes rather complex, and I wondered whether Overleaf could tackle it. These doubts proved to be wrong. I was quite impressed by the fast and correct typesetting on Overleaf servers.

The installation is rather complete. Currently the user can choose between *pdflatex*, *xelatex*, and *lualatex*, and the site has TEX Live installations from 2014 to 2019. Both *bibtex* and *biber* are supported.

In general, Overleaf has solved the problem that has always bugged novice TEX users: administration-free maintenance of the installation.

5 Comments system

In the old times manuscripts and books had large margins because coauthors, editors, and sometimes even readers used them for metacomments (Figure 2).

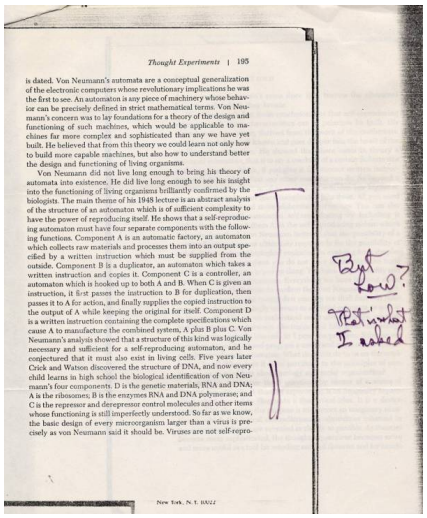

Figure 2: Annotations by Linus Pauling made to a passage in the book *Disturbing the Universe*, by Freeman Dyson, from `http://scarc.library.oregonstate.edu/coll/pauling/dna/notes/unpb17.1-automata.html`

When word processors appeared, they featured "comments": a user could click on a document and insert a special text, which was not part of the main flow, but a note for other authors. Web collaboration tools kept this feature and added integration with e-mails, messaging and task recording software; the comment could be "assigned" to a coauthor as a new task, and the assignee would be notified electronically with the corresponding record made in the task list.

TEX (or LATEX) by itself lacks this sophistication. Of course, one can add comments in the code using, for example, the percent sign convention, as in

```
\begin{equation}
 2\times 2 = 5    %%% Are you sure?
\end{equation}
```

Unfortunately, sometimes these comments are overlooked by coauthors. For better visibility, LATEX packages like *todo* may typeset them in the PDF output, for example, as marginal paragraphs.

The use of version control servers like GitHub allows one to employ a modern issue-tracking system which has all the functionality of the integrated comments, including notifications, assignments, etc. Moreover, the issues become part of the version control archive, so one can look at the history and establish that the given change in the manuscript was made by author A as a response to the comment made by author B — complete with the dated history of all responses.

For a user accustomed to these goodies, the Overleaf system of comments seems to be rather old-fashioned. Overleaf allows the users to create

a "comment" in the style of early word processors. These comments can be answered and "resolved". However, they are not shown in the PDF, cannot be addressed to a specific coauthor, and are not integrated with emails or the version history. While relatively easy to use, these comments are probably not adequate for a sophisticated user.

6 Overleaf version control system

An important feature of a collaboration tool is the ability to establish who wrote what, when and why. This is one of the tasks for a version control system. A version control system keeps track of the changes in the files, noting their authorship, dates, and, through commit messages, the reason for change. Ideally, it allows easy mixing of versions; a good program should understand a command like "Please delete all changes made in June, but keep the ones made in July".

The native version control system in Overleaf can do some of these tasks. It allows the user to compare the states of the document at the different stages of editing, find the changes, establish their authors and download the files at a previous version. This makes the work with Overleaf easier than with typical office software.

Unfortunately, the Overleaf version control system lacks branching capabilities. This means that major changes in a project require copying and renaming files — something that a version control system is intended to prevent. I thought initially that only the Overleaf interface to version control was deficient, and I could use branching in my *git* repository, and then synchronize with the Overleaf one. I was wrong; sadly, Overleaf allows only one branch to be uploaded to its repository.

Overleaf also lacks tags, and does not allow one to push them from the local *git* repository.

Reverting to a previous version is not a self-evident task in Overleaf. One of the coauthors inadvertently deleted a large portion of our TEX file in Overleaf. We found that the simplest way to recover was to use the tools provided by *git* on a local repository.

I was able to perform non-trivial merging of two different versions only by downloading them to the local computer, and using the tools there.

Overleaf version control system has "labels". They are mapped into *commit messages* rather than *tags*. They are also not enforced, so most changes in the files are not commented (in contrast, most version control systems refuse to accept commits without comments).

Of course, many *git* users complain it is too complex. The simplified version control of Overleaf might be a clever decision. While a sophisticated version control system allows an advanced user to perform many non-trivial tasks, it is indeed complex exactly because it has a rich feature set. An ideal system would be the one that works simply for a novice, but can do complex things. Clearly, designing such a system is not a trivial task. On the other hand, Overleaf solved the highly non-trivial problem of creating an intuitive interface for TEX — maybe it can create an intuitive interface for *git* as well?

7 Conclusions

Overleaf as a collaboration platform is a viable solution for coauthoring scientific manuscripts. Its strong features are a good access control model and underlying TEX installation. On the other hand, its system of metacomments can be improved, and its version control system is probably too simplistic.

Acknowledgments and disclaimers

I am deeply indebted to my colleagues who allowed me to observe their work with Overleaf.

Being a TUG officer, I acknowledge the generosity of Overleaf in their support of the organization through their institutional membership and several donations, including their help with TUG 2020, especially the LATEX workshop held there. The work described in this paper was done using a Professional level account, paid by Chan Zuckerberg Initiative.

The opinions in this article belong to me, and do not necessarily reflect the opinions of TUG, Chan Zuckerberg Initiative or George Mason University.

References

[1] G. Huber, M. Kamb, K. Kawagoe, L. Li, B. Veytsman, D. Yllanes, and D. Zigmond. A minimal model for household effects in epidemics. *medRxiv*, 2020. doi:10.1101/2020.07.09.20150227

[2] S. Satish, Z. Yao, A. Drozdov, and B. Veytsman. [A paper under blind review]. In *[Not disclosed]*, 2020. We were asked to suppress all information about the paper until it is reviewed.

◇ Boris Veytsman
Chan Zuckerberg Initiative
& George Mason University
& TEX Users Group
borisv (at) lk dot net