Collaborative Authoring

Joe Thurbon | 29/11/10 | V1.0

Abstract

This document discusses the pros and cons of different approaches to collaborative authoring. The document concludes with a checklist of questions that a team should be able to answer before embarking on a collaborative authoring endeavour. This document attempts to arm the reader with possible answers to these questions, and the knowledge to choose the most appropriate answer for their circumstances. The questions are:

1. What are the editors that the team will use?, and
   a. Do they support merging of work?
2. How often do we expect that people will want to edit the same document at the same time?
3. What services does the department/faculty/university provide for:
   a. backup;
   b. network drives;
   c. versioning repositories like SharePoint or Subversion.
4. What processes should we put in place to make sure that:
   a. copies never get lost;
   b. copies can be shared between authors;
   c. all authors know what the ‘latest version’ is at all times;
   d. all authors know when it is safe to edit a copy of a document.
Table of Contents

Introduction ......................................................................................................................... 3
Audience ............................................................................................................................... 3
Why read this document .................................................................................................... 3
Terminology and a motivating story .................................................................................... 3
Definitions ........................................................................................................................... 4
The basic workflow – sequential editing ............................................................................. 5
Email as a sharing tool ......................................................................................................... 6
Recommendations ............................................................................................................... 6
A shared drive ...................................................................................................................... 7
Recommendations ............................................................................................................... 8
A Versioning Repository ..................................................................................................... 9
Lock – Edit – Unlock .......................................................................................................... 9
Recommendations ............................................................................................................... 10
Update – Edit – Commit ....................................................................................................... 11
Recommendations ............................................................................................................... 12
Combination Systems ....................................................................................................... 13
Summary and Checklist .................................................................................................... 14
Introduction

Research is often a collaborative endeavour. Whether writing documents in Microsoft Word, performing an analysis in NVivo, or wrangling figures in Microsoft Excel, many documents are written by teams. This team-based creation of documents is called collaborative authoring.

While the benefits of collaborative authoring are obvious, there are also several pitfalls. This document

- provides definitions and explanations that can be used to describe the process of collaborative authoring,
- outlines some of the pitfalls of collaborative authoring, especially those which result in the loss of work, and
- describes some ways to avoid those pitfalls, making collaborative authoring more effective.

Audience

This document is aimed at researchers who author documents collaboratively, and in particular those who are not from an IT background. That is, it is aimed at researchers who may have experienced some of the pitfalls of collaborative editing, wish to avoid them in the future, and are unsure of the tools and practices that are available to them.

Why read this document

At its best, collaborative authoring allows people with diverse skills to create a single cohesive document quickly. At its worst, collaborative authoring results in significant re-work, lost work, and frustration.

The lost work can take many forms: people’s files get lost in email, people save over other people’s work, multiple people edit the same document simultaneously and their work conflicts with each other.

This document will arm authors who are about to work collaboratively with the knowledge to structure their collaboration so that these pitfalls can be avoided.

There are many ways to structure the workflows of a team around collaborative authoring. The remaining sections of this document detail some of the common ones. Before that, we conclude this section with a motivating story to introduce the terminology used in this document.

Terminology and a motivating story

A group of five authors wish to collaborate to write a document. That document’s contents will change over time: it will be argued over, be edited and reviewed, and eventually be published. Throughout this time, it remains the same document.

As a first step in writing the document, one of the authors opens up an editor and starts typing. After some brief period, the author saves his/her work to a file, say C:\Temp\Doco.doc. This file is the document’s first copy.
The author keeps working on that copy until they are ready to have the other authors review it. The team has collectively decided that important copies of the document will be saved on a shared drive. We’ll call that shared drive a repository for copies of the document.

The author saves their copy to S:\Repository\Doco-v1.doc. There are now two copies of the document: one in C:\Temp, and one in S:\Repository. That new copy in the repository is designated the draft version of the document.

To facilitate review, the author emails the document to the four other authors, and thus spring into existence four new copies.

**Definitions**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>An author is any person who uses an editor to modify or create a copy of a document.</td>
</tr>
<tr>
<td>Copy</td>
<td>A copy is a file whose contents encode a document at a particular stage in its creation. Each copy has a particular format, for example: MS Word, PDF, Jpeg, and MP3.</td>
</tr>
<tr>
<td>Document</td>
<td>A document is an idealised conceptual entity that has a topic and some contents.</td>
</tr>
<tr>
<td>Editor</td>
<td>An editor is a program that is used to create or manipulate copies of a document. Some editors are made specifically for editing particular kinds of documents, e.g. MS Word, whereas others are very general, e.g. text editors or Photoshop.</td>
</tr>
<tr>
<td>Repository</td>
<td>A repository is a piece of software that is used to house copies of documents. They range from the local hard drive, to thumb drives through to full fledged version-control systems. The distinguishing feature of a repository is that people who put their copies there expect those copies to remain there.</td>
</tr>
<tr>
<td>Version</td>
<td>A version is a copy of a document that has been earmarked as canonical. Common versioning idioms are to number versions (e.g. 0.1, 2.0), to use a small list of terms (e.g. draft, camera-ready, published), or to use ad-hoc descriptions (e.g. ‘Joe’s edits of Cameron’s reviews’).</td>
</tr>
</tbody>
</table>
The basic workflow – sequential editing

The most basic workflow for collaborative authoring is that authors take turns in editing the document, as shown in Figure 1. From an initial copy, Joe makes edits to create a second copy. From this second copy, Clare makes edits to make a third copy, which is then edited by Elaine.

![Figure 1: A linear document workflow](image)

Even for this simple scenario, there are many questions that must be answered.

1. At any given point in time, how many copies of a document should exist, and where?
2. At any given point in time, how does an author know it is safe to work on a document?
3. At any given point in time, which is the canonical copy of the document?
4. Once an author has decided to edit a copy of a document, how does the author obtain that copy to edit?

There are no hard and fast answers to any of these questions. Picking answers will make some tasks easier, and some harder.

For example, in the scenario outlined above, possible answers to question one include: just one, where each of the authors saves over the last author’s work; four, where every copy of the document that gets made is kept in perpetuity; two, just the original and the latest version are kept.

Answers to questions two and three can range from using conventions like a physical token or document naming conventions, through to using a repository that enforces one-at-a-time writing.

Answers to question four can range from using a single computer at which authors take turns to edit, through using email or a network drive, through to using a full blown document repository.
Email as a sharing tool

Figure 2 shows the most common ad-hoc solution to answering some of these questions.

![Diagram of email sharing process](image)

**Figure 2: Email as a distribution method**

First Joe makes his edits, and then emails a copy to Clare. Then Clare makes her edits, and emails a copy to Elaine. This method has a few compelling advantages. It is clear to Clare when she can start work: it is when Joe has emailed her the document. It also answers the question about how each individual shares their work with other people.

It has a few major failings, however. Most importantly, it becomes very difficult to simultaneously determine both of:

- which is the canonical copy of the document?
- when can I start work?

In the scenario shown above, it is easy for Elaine to answer “When can I start work?” but impossible for Joe to know “What is the canonical version of the document?” In an alternate scenario, where everyone emails everyone else every copy of the document every time a change is made, it becomes possible to know “What is the canonical copy of the document?” but very difficult to know “When can I start work?”

Completely unaddressed by this approach are questions like: “what copies should remain in existence in the long term?” and “where do I go looking for old copies of the document?”

**Recommendations**

When using this workflow, you should

1. Define a location that houses canonical copies of a document, and ensure that that location is backed up (i.e. establish a repository).
2. Have some sort of process in place to:
   
   a. Announce when new canonical copies are added to the repository
   b. Arbitrate who is the current editor of the canonical copy, to ensure that only one person edits it at a time.
A shared drive

The easiest way to nominate a repository for canonical copies of a document is to use a network or USB drive as a shared repository for documents. This approach is shown in Figure 3.

In this scenario, a team assigns a shared disk (either removable or network) as a repository to store canonical copies of documents. Some teams further mandate that all work happens directly on the shared drive – that is, no local copies are made. This approach has the advantage that sharing between team members is very easy, and also that the shared drive integrates seamlessly with the team’s environment.

Assuming that the shared disk is backed up, the main way that data loss can occur is by overwriting one another’s work, as shown in Figure 4.

Either Clare will save her work over Joe’s copy, in which case Joe’s work is lost, or, Clare and Joe will both have copies on the shared drive. In the latter case either:

- Joe’s work will be lost because Clare’s copy becomes the canonical copy, and no-one will ever know Joe’s work was done or;
- someone will have to expend the effort of merging Joe’s work with Clare’s.
Recommendations

When using this workflow, you should

1. Have some sort of process in place to:
   a. announce when new canonical copies are added to the repository;
   b. arbitrate who is the current editor of the canonical copy, to ensure that only one
      person edits it at a time

Note that the policy of “no local copies”, plus arbitrating access to the shared drive means that work
 can never be lost. It also means that work can never be simultaneous.
A Versioning Repository

To address this final form of lost work, some teams employ a versioning repository, also known as a revision control system. There are general offerings, such as Subversion, CVS and Git, as well as document repositories such as SharePoint, Alfresco and MS Live.

The two main pieces of functionality that a versioning repository adds to the above scenarios are:

- repositories manage versions on the behalf of authors, ensuring any copy of the document which is lodged in the repository is kept in perpetuity; and
- repositories arbitrate access to copies of documents in a managed way, so that potential loss-of-work scenarios are either avoided or brought to light quickly.

The cost of these benefits are that in some cases, the day-to-day workflow of authors is made slightly more complex. Most versioning repositories mandate one of two workflows. They are documented below.

Lock – Edit – Unlock

The Lock-Edit-Unock workflow described below. Consider a situation where an author, Joe, wishes to edit a document. He takes the following steps

1. He informs the repository that he wished to edit the document. The repository confirms that no other author is currently editing the document, locks the document, and delivers Joe a copy of the document.
2. Joe edits his copy. During this time, if any other author attempts to lock the document for editing, they are told by the repository that Joe is editing the document.

![Diagram](image1)

3. When he is finished, Joe saves his changed copy of the document back into the repository. At this point, the document is unlocked. Both the version of the document which Joe began editing as well as the version of the document which was created by Joe are both held in the repository in perpetuity.

![Diagram](image2)

The most significant drawback of this workflow is that no concurrent work can be carried out. As such, this workflow is recommended for editors that have poor support for merging work. That is, editors like PhotoShop, Microsoft Excel, and to a lesser extent Microsoft Word make merging of work burdensome, so ensuring concurrent editing does not occur is typically encouraged.

There are many workarounds that allow some concurrent editing, however they vary depending on what repository is used, and as such they are outside the scope of this document.

**Recommendations**

Using this workflow, one should:

1. attempt to minimise the time that documents are locked. This is especially true around weekends and periods of leave;
2. ensure that there are easy lines of communication between authors so that locked documents can be unlocked easily.
Update – Edit – Commit

The Update-Edit-Commit workflow is relatively complicated, and should only be used when either:

- the editors being used support merging; or
- authors have decided that the benefits of concurrent editing outweighs the burden of doing manual re-work.

Where the Lock-Edit-Unlock workflow attempts to manage concurrent access at the beginning of each editing session, the Update-Edit-Commit workflow attempts to manage concurrent editing at the end of each editing session.

Consider the following, where Elaine and Luc edit a document at the same time.

1. Elaine first asks the repository for a copy of the latest version of the document. The repository gives her a copy. Elaine begins editing her copy, taking some time to do so.

2. Luc asks the repository for a copy of the latest version of the document. Since Elaine is still editing, Luc gets a copy of the same version that Elaine did. Luc begins editing his copy.

3. Elaine finishes editing, and informs to repository that she has done so. The repository creates a new version of the document with Elaine’s copy.
4. Luc finishes editing. When he attempts to push his change back into the repository, the repository informs him that Elaine has already changed the version that he was working on.

5. Luc is given a copy of Elaine’s changes, and Luc must then merge his and Elaine’s work together.

6. Once that merge is complete, Luc can add his copy back into the repository.

This is the most flexible and permissive workflow. It also ensures that work is never lost. The benefit of this approach is that it is possible for many authors to work simultaneously, and merge their changes at the end of their work.

The most obvious drawback of this approach is that it relies on having good support for merging after work has been completed.

Recommendations
If you choose to use this workflow, then:

- understand the merging options that are available to you, and trial them with your team;
- understand that this process merely makes explicit the access arbitration and merging that is implicit in the above processes
- choose a versioning repository that makes the right usability/flexibility tradeoffs for your team.
Combination Systems

There are some systems that merge one or more pieces of the system. The most common merging is the editor and repository. Systems that do this are:

- Wikis
- Microsoft Live
- Google Docs
- NVivo server 9

These systems are great for ease of use. They have the drawback that, if your team needs to use more than one editor (e.g. MS Excel and NVivo) then the canonical versions of your documents will end up in two repositories.
Summary and Checklist

This document has discussed the pros and cons of different approaches to collaborative authoring. We finish with a series of questions that a team should use before embarking on a collaborative authoring endeavour – the answers to these questions should guide you in choosing which of the above approaches to consider.

Your Intersect eRA is, of course, available to discuss these issues.

5. What are the editors that the team will use?, and
   a. Do they support merging of work?

6. How often do we expect that people will want to edit the same document at the same time?

7. What services does the department/faculty/university provide for:
   a. backup;
   b. network drives;
   c. versioning repositories like SharePoint or Subversion.

8. What processes should we put in place to make sure that:
   a. copies never get lost;
   b. copies can be shared between authors;
   c. all authors know what the ‘latest version’ is at all times;
   d. all authors know when it is safe to edit a copy of a document.

---

<table>
<thead>
<tr>
<th>Version No.</th>
<th>Revision Date</th>
<th>Summary of Changes</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>29.11.2010</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---