Version control for ebook publishing

Pierre Thierry

July 2013

Contents

1 Introduction

Version control (VC) has been a critical tool for many software developpers, to the point that some features offered by version control systems (VCS) have actually enabled new and improved workflows, both on a technical and social level. VCS, and more recently decentralized VCS (DVCS), have brought major improvements in safety, quality assurance as well as flexibility in team organization and project management.

It is then all the more surprising that the programming community never really managed to disseminate this technology to the general public. Apart from programmers themselves, the only other community actually using VC on a large scale is the Wiki users, mostly because wikis are a tool originating in the programmer community.

Although mostly restricted to software development currently, VC could be useful at each and every stage of ebook writing and publishing. This document briefly presents various use cases of VC for the different stages of creation and publication of an ebook, based on experience with novel authors and researchers in humanities. The requirements of those use cases are also compared with the use of VC in software development, to determine where existing tools can be readily used and where tools need to be extended or created from scratch.

2 Version control use cases for ebooks

The use cases mostly follow the life cycle of a book from writing to publication, but it is important to realize that, as far as life cycle is concerned, where paper books cannot but follow what is called in software development methodology a waterfall model, with each stage needing total completion of the previous stage to take place, ebooks have brought the possibility of a more agile development model, with stages intertwined or the development cycle partially taking place again on several occasions.

Actually, the ease with which ebooks can be made by even inexperienced authors and technicians may make it all the more necessary to be able to publish corrected versions quickly and frequently. Ebook publishers, for better or worse, may embrace free software's motto "release early, release often". VC is specifically useful in that regard.

2.1 Authoring

As a large portion of books are essentially the work of one person and evolve in a rather linear fashion, it may seem overkill to use a tool like VC to track the evolution of a manuscript. But the reality is that even a single author whose modifications are mostly additions to their text with occasional rewritings will usually feel the need to record snapshots of his work at regular intervals. Doing this manually has been known to be error-prone, time- and space-consuming.

In most cases, authoring only needs support for a linear history. But even at this early stage, many books already involve a team, sometimes even working concurrently on the same places in the book[1], which quickly

[1] A canonical example is the writing of a technical or process documentation, where some people might be charged with modification across the whole book to apply some content policy while others might still be writing the documentation itself.

leads to loss of either data or productivity, as the simple solutions to avoid data losses involve making concurrent modifications impossible.

Whereas a single author just need the ability to create and retrieve versions in a linear history, concurrent editing actually makes use of most basic features of VC, including branching, patch revision and conflict resolution.

2.2 Proofreading

Proofreading obviously benefits from the ability to work concurrently on the same document and to automatically represent modifications to be reviewed. The reliability of document and change transmission also sees a vast increase.

Existing authoring tool may provide adequate change revision, like text editors like Word or Writer do, but they only support a linear history embedded in a single file. Not only branches but also concurrent edition are out of the question, thus greatly impairing productivity[2].

2.3 Ebook finalization

In the waterfall model, the stage where someone takes the authored text to turn it in a proper ebook doesn't really benefit much from VC apart for easy transmission and backup.

But the nature of ebook publishing makes it less likely that a book will not see a number of revisions, which would mean, without being able to merge the work of creating the ebook file and the errata, that this stage would need to be redone partially for each new release, almost entirely depending on the workflow and the file formats used by contributors of the previous stages.

2.4 Publishing

Even while least affected by VC, the publishing end of a book creation process might still benefit from the improved manageability of concurrent branches and successive versions of a book.

The uniform process for change storage and transmission accross the whole workflow would actually mean that a publisher can now closely follow each step of the creation and accurately monitor each contribution, almost in real-time.

[2]Usually, as letting go of concurrent editing is not really an option, an author will actually use embedded change revision only to spot changes, not to integrate them

2.5 User customization

A book continues to be seen mostly as a black box, even as they now are just files with a standardized structure that simple took make editable. The use of DRM by publishers may strenghten this black box illusion.

But new publishing models appear that take a more liberal approach to the content of an ebook. Most prominent are the open access movement in scientific research and the advent of crowdfunding that ends up with the release of a book under an open license or in the public domain, as envisioned in the Street Music Performer Protocol by Bruce Schneier.

These open ebooks open many possibilities in content enriching and user customization that are quickly guaranteed to create a mess of conflicting and ill-identified versions. This possible problem finds a cheap, easy and scalable solution with open and standard VC.

3 Limitations of existing tools

Because VCS is only really used in programming circles, its tools suffer from a cruel inadaptation to other audiences. Even graphical interfaces rightly emphasize on flexibility and on giving access to every possible feature of the tool, whereas most contributors to an ebook would need a radically simplified user interface (UI) with a guided workflow.

But if writing another UI should be a task easy enough, the most critical work would be to provide a patch revision and conflit resolution interface that is adapted to text authoring, as opposed to source code authoring. The current default patch revision for source code is line-based, whereas text would need either word-based or structure-based patches, indicating which words have seen modifications or which chapter, section or paragraph of the book.

Encouraging is the fact that some of the widely deployed VCS tools today provide both extreme flexibility and are tailored to be extended with alternative UIs, patch revisions and conflict resolutions, which means that the ebook industry should be able to adopt VC with modest tool development effort and a continued support from the current VCS providers.

4 Conclusion

VC is probably a vital missing element of the ebook industry right now, explained in part by the lack of tools adapted to its craft. But a rather

limited and progressive community effort should make it possible to quickly bridge the gap in terms of available tools and bring ebook publishing to a brand new level.

The important intersection between the software development and ebook publishing world should also make it easy to conduct experiments of VC deployment in the workflow with increasing scales.