Collaborative Authoring Requires Advanced Change Management

Ethan Munson
Department of EECS
University of Wisconsin-Milwaukee
Milwaukee, WI, USA
munson@uwm.edu

ABSTRACT

Collaborative document authoring is a pervasive activity of modern life. The obvious examples are jointly authored works of scholarship, literature, and journalism. But collaboration can also be found in many work domains: doctors collaborate on the creation of a patient's chart; lawmakers and their aides collaborate on legislation; opposing lawyers "collaborate" on a legal settlement; editors collaborate with authors to correct or improve manuscripts.

In addition, while software engineers rarely give their work the name "collaborative authoring," that is precisely what it is. Because of this and because it is their nature to make better tools, software engineers have created powerful and robust programs to support collaboration, such as version control and build systems. These tools allow software engineers to share documents, divide them into complicated subdocuments, edit them in parallel, merge editing changes semi-automatically, and recombine the subdocuments into a cohesive and correct whole.

It is the thesis of this talk that collaborating authors of natural language documents need the same kinds of tools that software engineers take for granted. In fact, because of important domain differences, people collaborating on a natural language document need better tools than software engineers. One key difference is that natural language documents lack clear standards of correctness. Where software engineers can use compiler errors and test results to validate the merged changes of multiple authors, natural language authors, especially for creative literature and poetry, have few equivalent tools. Another difference is that some collaborators (see the legal example above) may not trust each other, so even changes that do not generate a conflict still must be validated manually by the other collaborators. Finally, natural language authors and editors are not programmers, so they need solutions with interfaces that are accessible to a non-technical audience.

The document engineering research community has been working on this problem domain for a considerable time, particularly in the area of version control of XML documents, which is the primary representation for modern office documents. Key research results have included a variety of schemes for merging and patching XML document versions, advances in formalizing document deltas, a scheme for controlling author access to the various sections of a document, and demonstrations that authors can edit a document simultaneously over the Internet without locking mechanisms.

I will argue that document systems must adopt the full range of version management tools used by software engineers, including full branch-and-merge versioning, but then must extend those tools and simplify their use. Naive users cannot be expected to maintain version repositories or even to follow a protocol rigorously. We need approaches to sharing documents and merging their changes that are highly automated, that recognize the difference between formatting and content, that permit fine-grained access control, and that help users understand the provenance of changes so that responsibility is correctly assigned. My laboratory has taken initial steps towards this vision with efficient differencing and three-way merging and by introducing *Version-Aware Documents* that carry the full version history in an office document file. But the long term vision requires better user interfaces, better algorithms and schemes for identifying document differences, and probably, lightweight artificial intelligence to improve merging of conflicting edits.

Once office documents have this kind of advanced change management, it is easy to picture extending the paradigm to specific domains. What if a patient's medical history was viewed as a series of "versions" of the patient? What if automated tools could easily identify subtle changes that one party is trying to "sneak into" a new law or a contract? What if the same techniques that automate merging of conflicting human-generated edits could be used to correct document analysis errors?

Categories and Subject Descriptors

I.7.1 [**Document and Text Processing**]: Document and Text Editing—*Version control; Document management*

DChanges 2013, September 10th, 2013, Florence, Italy.
ceur-ws.org Volume 1008, http://ceur-ws.org/Vol-1008/keynote.pdf .