The Next Millennium Document Format

Svante Schubert
Berlin, Germany
SvanteSchubert@apache.org

ABSTRACT

Most of today's leading document formats have their roots in the eighties. Their design was built upon requirements of these days: to represent the document state on one single machine or to exchange a document by floppy disc or modem. Often designed for a single purpose far narrower than their current usage. New features were often accomplished by workarounds. For example, change-tracking of any office format does not track a defined interoperable change. Only the earlier state of the changed area is stored, to be swapped back in case of rejection.

Nowadays, with the rise of mobile devices, online collaboration is ubiquitous and creates challenges when dealing with documents designed for an environment from the eighties.

In this paper, we lay out a concept how to evolve a new document format that allows not only collaboration, but responsiveness and interoperability by design.

CCS CONCEPTS

• I.7.1 [**Document and Text Processing**]: Document and Text Editing – Version control;

• I.7.2 [**XML**]: Change Control – *merge, change-tracking, versioning;*

• C.2.4 [**Computer Communication Networks**]: Distributed Systems – *Distributed applications.*

KEYWORDS

Document versioning, Document synchronization, Collaborative editing, Document repositories deduplication, Interoperable Layout, Agile standardization

ACM Reference format:

Svante Schubert. 2019. The Next Millennium Document Format. In *Proceedings of ACM Document Engineering conference (DocEng 2019). ACM, New York, NY, USA, 4 pages.* https://doi.org/10.1145/3342558.3345419

1 Introduction

Our vision is the evolvement of a new document format that fulfills the majority of today's user requirements by embracing three major concepts simultaneously by design:

1. Parallel Editing (real-time and offline).
2. Visualization with layout fidelity.
3. Interoperability and openness by agile standardization.

2 Parallel Editing (real-time and offline)

Our goal is to allow multiple, different editor applications to work on one rich text document (real-time and offline) and have all changes being merged into one result document in the end.

Our first design decision is to simplify the system by exchanging user changes instead of documents between applications. This drops the complexity to re-identify the changes from the document [1,2]. Compared with Software Development, the exchange of documents between editors is about as efficient as if software developers would zip their source code repositories and exchange those for collaboration.

Changes (or operations) can be sent alone as they are referencing to their point of change within the document being made at a certain time and document state. Such a point is often moved when another user has added or delete content before this point. Therefore, the position of operations is transformed during merge by the concept of Operational Transformation (OT) [3,4]. But despite the success of OT, an interoperable exchange of user operations between different applications is still not envisaged, as currently only the exchange of full documents is being standardized - like HTML, ODF, OOXML, DocBook.
Still there is the possibility to standardize collaboration on user changes, as we observe the existence of the same semantic document model among all document formats, consisting of entities such as a paragraph, character, table, image. For a specific format we observe even the existence of similar user changes working upon one semantic document model. But neither the semantic entities nor their user changes of "adding", "modifying" and "deleting" were ever specified in a standard.
Our second design decision is to simplify the system by relying on operations reflecting changes on the semantic document model[1].

[1] The standard EN16931 on electronic invoicing developed from European standardization organization "CEN/TC 434 - Electronic invoicing" defines the semantic data model of an electronic invoice and binds the semantic model to multiple XML invoice formats.

This higher-level semantic document model abstracts from the implementation details of the underlying syntax model [5]. For instance, a single user change - as changing the page layout on a paragraph - might result into four different XML changes within two XML files of a zipped ODF document. Every single semantic user change is transforming a document from one valid document state to another and might consists of multiple syntax changes.

Our third design decision results from the fact that the final document is equivalent to the sum of all changes the user has made. In case of a single user, the document is equivalent of a timely ordered list of her changes. The user change (or operation) is similar to a commit in software development, which often consists of multiple line changes. Therefore, the design idea is that every editor - or collaborative editing (CE) system – is working like a software developer on a local branch [6,7].

Branches allow us the maintenance of multiple releases of the same document, keeping the development of major features isolated, and trying out experimental changes without affecting the main line. The work process for collaboration on branches is automatic, but otherwise similar to GIT [8,9,10]

Foremost, the concept of branches allows to simplify the synchronization of operations to a process similar to a GIT rebase[2].

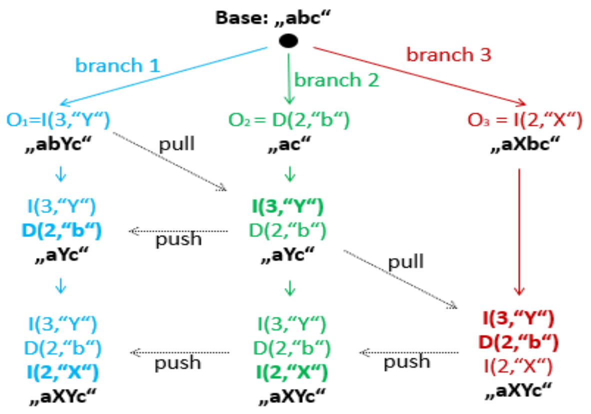

Figure. 1: Merging three operations using the GIT rebase concept

The OT being involved is best described by an easier example first: We know that a document is equivalent to the list of user changes. Still different lists of user changes might result into the same document. Aside of removing couples of inverse operations, such as adding and deleting the same entity, such lists are being normalized by atomic steps of switching two adjacent changes. Switching two changes on the timely ordered list involves OT (Figure 2). By switching the execution order of two adjacent changes their position might be transformed in order that the combined result is still the same. The document state is therefore not being changed. It is therefore important for operation design that two adjacent operations can be switched without further knowledge of the document. Not all operations can be switched as the lifecycle of entities has to be respected: first occurs “insert”, afterwards sometimes “modify” and “delete” only at the end.

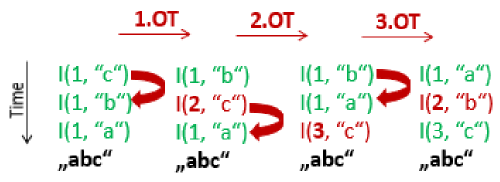

Figure. 2: User change list normalization in three OT steps

OT during a rebase is a bit different than OT during normalization, as the pulled changes are never being altered. As pulled changes are the shared history being accepted by the pulling and had occurred earlier. Like during GIT rebase the transformed own changes are being appended. Each pulled change is checked if it influences a local new change by OT. Similar as if the pulled operations were temporally being appended (would occur after local changes), but then moved one by one through the local new operations while applying OT (influencing the local changes) until they passed them and finally occur before them. Other minor design decisions in contrary to former OT are the representation of a rich document (as ODF) as a tree data structure. Analog to the underlying XML syntax trees of the XML files within its ZIP file oppose to an array as in OT. We assume advantages of less OT activity and easier move of subtrees. Another minor design decision is that in our data model a semantic object is identified by its position in the semantic tree starting counting by one as in XML in contrary to 0 in OT. For instance, the second character in the thousands semantic object being a paragraph might be written as „/1000/2“. Implicit positions are far more powerful than IDs. For example, if an author is publishing read-only her book and receiving a GitHub-like „pull request“[3] of a reader to correct a typo by deleting the character „/1000/2“. Without implicit numbering the author would be forced to add an ID at any possible place, increasing document-size. Similar as in GIT the collaboration of any number of users is possible by synching automatically two users at the time. Although there is no ground truth of one branch by definition, it could be established by choosing a centralized distributed GIT workflow[4]. All users (branches) have in common to be able to go back to a document state they once shared. We are aware that the GIT approach of rebasing changes from multiple user cannot be free of conflicts and is not meant to be. For instance, a conflict might be: One user changes a table cell, the other deletes the table. But conflicts do not always have to be resolved by the user, for instance resolving changes of multiple real-time users might rely on a heuristic to resolve conflicts on all clients in a consistent way. Still if the document is of a high value it might in the interest of the document owners that only humans resolve conflicts and the complete document history is being archived. In contrary to GIT a document state is not represented by the hashing the syntax of the document, as XML can vary too easily representing the same document, but instead of using XML syntax the semantic tree is being used (semantic document hash).

[2] https://git-scm.com/docs/git-rebase

[3] https://help.github.com/en/articles/about-pull-requests

3 Visualization with Layout Fidelity

Whenever the layout of a paged document – or the backward compatibility to paper - is of importance, for instance when during a meeting some content is being referred by page number or in case business documents with high-fidelity corporate identity design are being sent to customers; most document formats can only guarantee the same layout when the same application on the same platform is being used by all participants or by sending PDF, graphics or similar formats, which are close to the layout but far from former structure and user semantic and by this difficult to edit.

The technical reason for this drawback of document formats viewed by page layout (e.g. office formats such as OOXML & ODF or HTML) lies in the transformation of the syntax of their document to a visible representation for the user. In this transformation, called page layout, the application is positioning viewable content to the given area of pages. As the PDF format is very close the final layout, it provides in general interoperable layout with stability across all applications and platforms. The natural drawback of PDF for this layout stability, is the difficulty of editing and adjusting this layout according to user changes [11]. What all document specifications are missing is a definition of the layout from text to lines. The likely reason for this is that performant operating system functions are being called to render the text in a platform dependent manner, instead of providing interoperable rendering of text into lines on pages for all platforms. (La)TeX platform independence is an exception, but is not fully supported by WYSIWYG editors required for usability of a broad audience. Still there is much to learn and overtake from (La)TeX [12].

Our goal is that if there would be a new document format it should provide not only the successful responsive design to the device's viewport to allow usage across various screen sizes similar to existing HTML, but as well should offer layout fidelity for rendering on page size (page layout) even across various applications and platforms to fulfill this basic need of layout fidelity.

4 Interoperability and Openness by Agile Standardization

Our goal is that documents - as backbone of our societies knowledge - should be as open and reusable (interoperable) as much as possible for users. The common way to achieve this is by an Open Standard being the blueprint for global implementations.

Many standard organizations do exist worldwide, but the highest impact have those supported by governments. The most important is the International Organization for Standardization (ISO) with their national partner organizations such as DIN for Germany or their European counterpart (CEN). Only the government supported standards are able to be required by law, for instance the usage of ODF format as office format for the use across government was mandated by Great Britain[5]. There is an incentive for big companies resulting from this, which was likely the reason why the ISO standardization of ODF triggered the existence of OOXML. But software development had become very fast and agile in the past decades, while the standardization of software artefacts (file formats) still follows long distribution cycles. There is a timely discrepancy between software sprints 1-2 weeks in which new releases are being delivered and the release cycles of their blueprints of 6 to 12 months at best.

Reason for the long cycles are that errata of standards are still being written manually and not generated from the changed standard documents, the absence of automated tests of the specification and that all changes have to be reviewed and voted by all international partner organizations. Another reason for the slowness of creation, but even more for the slowness of adoption the standard is that no opensource tests for implementations exist at standardization level. Every implementation has to create their own tests and there is no default feature comparison tool like the HTML ACID tests[6]. Also, complicated for document standards are the high number of new versions. Does an application have to implement all versions? There is no transformation of existing documents from the previous version of the standard to the next version. If so, a chain of transformations could transform every version of the document (software input) to the latest version, allowing new implementors only to work with a single standard – the latest one. Currently, a document standard with many versions becomes more difficult to implement for new applications with every new version or old documents might become no longer accessible by new applications.

Most of all, there is a gap between the information access available during standardization and the information the buyer of an ISO specification receives. A good example for the existing problems is the CEN standardization of the EU file format on e-invoice (EN16931)[7]. The standard provides the blueprint for software supporting two XML file formats, but instead of providing machine readable artefacts to generate source code from, only PDF or paper is being given out. We joined the German National Body DIN to get access to the office documents and extracted the main data by creating an open source tool[8] to allow editors to validation the data written within the specification. By providing not only pre- and postconditions but also example algorithms with the specification source code might be generated. The ultimate goal should be the generation of a reference software directly from the standard as such automation would guarantee a standard not being underspecified and would accelerate the speed of implementation in a never existing manner. In regard of parallel document collaboration not only the XML, but also its binding to semantic entities (groups of XML nodes) and the allowed user changes (change API) with a generic

[4] https://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows

[5] https://www.gov.uk/government/news/open-document-formats-selected-to-meet-user-needs

[6] http://acid3.acidtests.org/

[7] https://standards.cen.eu/dyn/www/f?p=204:32:0::::FSP_ORG_ID,FSP_LANG_ID:1883209,25&cs=126F1BDBC8D6D6141F550EB578B4A9CF4

[8] https://github.com/svanteschubert/en16931-data-extractor

description how every user change will affect the XML (or underlying syntax) have to be provided.
In regard of stable layout, the mapping from text characters to lines have to be described, likely with offering default algorithms.

5 Current Progress and Future Work

The current progress and future work are threefold:

1. The work on a document design to support collaboration is the most advanced area. We are able to provide a transformation of ODF Text documents (ODT) to user changes (in JSON) with the upcoming ODF Toolkit version 1.0.0.[9] In addition, new user changes (in JSON) are being merged back into the ODT document. The next step will be a prototype to connect various WYSIWYG editors via ODF changes. The idea is to establish collaboration between editors of different file formats supporting different ODF feature sets. The following are being considered: LibreOffice[10] as leading ODF application, CKEditor5[11] as advanced HTML editor, VSCode[12] editor as one of the leading opensource software editing tools and/or Emacs[13] as most famous text editor. For instance, Emacs as text editor would emulate a paragraph by a line and allows only text. All unknown features are meant to temporarily be stripped down before providing the supported operations to the application, but will be merged back when the user changes are being applied back to the original document. Therefore, would not break any full-featured office document during editing. The goal of this new prototype is to attract new ODF application into the OASIS ODF standardization, especially into the – currently hibernating subcommittee of „Advanced Document Collaboration“, which chosen the described concept over alternatives from Microsoft and DeltaXML [12] and is chaired by the author.
2. The idea of visualization with layout fidelity although discussed with implementors is merely a vision today.
3. The idea of interoperability and openness by agile standardization is in progress by working on the e-Invoice EU Standard (EN16931). The standard is provided only as PDF and paper to the end users. By participation it was possible to get access to the standard‘s editor office format version to create a data extraction to machine readable data. The data had been extracted to generate an opensource software based on an API aligned to the semantic model supporting both XML formats of EN16931. The aim is to show the possibilities of machine-readable specifications on EU level and improve the adoption of the EU e-invoice format.

[9] https://github.com/svanteschubert/odftoolkit/tree/odf-changes (currently) https://github.com/tdf/odftoolkit (soon)
[10] https://www.libreoffice.org/
[11] https://ckeditor.com/ckeditor-5/
[12] https://github.com/Microsoft/vscode/
[13] https://www.gnu.org/software/emacs/

ACKNOWLEDGMENTS

The Open Source software to transform an OpenDocument into its equivalent list of changes and to accept new changes was sponsored by the 2nd round of the PrototypeFund from the German Ministry of Research[14]. An Open Source software to implement the EN16931 e-Invoice standard and the extraction of data from the specification[15] was sponsored by the 4th round of the PrototypeFund from the German Ministry of Research[16].
Special thanks to Sebastian Roennau for the fruitful discussions and his comments.

REFERENCES

[1] Sebastian Rönnau, Geraint Philipp, and Uwe M. Borghoff. 2009. Efficient change control of XML documents. In Proceedings of the 9th ACM symposium on Document engineering (DocEng '09). ACM, New York, NY, USA, 3-12. DOI: https://doi.org/10.1145/1600193.1600197
[2] Sonja Maier and Sebastian Rönnau. 2015. A REST-based Document Model for Collaborative Editing of Documents. In Proceedings of the 3rd International Workshop on (Document) Changes: modeling, detection, storage and visualization (DChanges 2015). ACM, New York, NY, USA, 19-22. DOI: https://doi.org/10.1145/2881631.2881636
[3] C. A. Ellis and S. J. Gibbs. 1989. Concurrency control in groupware systems. In Proceedings of the 1989 ACM SIGMOD international conference on Management of data (SIGMOD '89), James Clifford, Bruce Lindsay, and David Maier (Eds.). ACM, New York, NY, USA, 399-407. DOI=http://dx.doi.org/10.1145/67544.66963
[4] David Sun, Steven Xia, Chengzheng Sun, and David Chen. Operational transformation for collaborative word processing. 2004. In Proceedings of the 2004 ACM conference on Computer supported cooperative work (CSCW '04). ACM, New York ,NY, USA, 437-446. DOI = http://dx.doi.org/10.1145/1031607.1031681.
[5] Exploiting single-user web applications for shared editing: a generic transformation approach. Matthias Heinrich, Franz Lehmann, Thomas Springer, and Martin Gaedke. 2012. In Proceedings of the 21st international conference on World Wide Web (WWW '12). ACM, New York, NY, USA, 1057-1066. DOI: https://doi.org/10.1145/2187836.2187978
[6] B. Appleton, S. P. Berczuk, R. Cabrera, and R. Orenstein. Streamed lines: Branching patterns for parallel software development. 1998.
[7] Shaun Phillips, Jonathan Sillito, and Rob Walker. 2011. Branching and merging: an investigation into current version control practices. In Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '11). ACM, New York, NY, USA, 9-15. DOI: https://doi.org/10.1145/1984642.1984645
[8] "A Short History of Git". Pro Git (2nd ed.). Apress. 2014. Archived from the original on 25 December 2015. Retrieved 26 December 2015.
[9] Santiago Perez De Rosso and Daniel Jackson. 2013. What's wrong with git?: a conceptual design analysis. In Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software (Onward! 2013). ACM, New York, NY, USA, 37-52. DOI=http://dx.doi.org/10.1145/2509578.2509584
[10] Santiago Perez De Rosso and Daniel Jackson. 2016. Purposes, concepts, misfits, and a redesign of git. SIGPLAN Not. 51, 10 (October 2016), 292-310. DOI: https://doi.org/10.1145/3022671.2984018
[11] Tamir Hassan. 2018. Towards a Universally Editable Portable Document Format. In Proceedings of the ACM Symposium on Document Engineering 2018 (DocEng '18). ACM, New York, NY, USA, Article 11, 4 pages. DOI: https://doi.org/10.1145/3209280.3229083
[12] Mittelbach, Frank. 2013. E-TEX: Guidelines for future TEX extensions — revisited. TUGboat, 34(1):47–63. ISSN 0896-3207.
[13] Svante Schubert, Sebastian Rönnau, and Patrick Durusau. 2014. Interoperable Document Collaboration. In Proceedings of the 2nd International Workshop on (Document) Changes: modeling, detection, storage and visualization (DChanges '14). ACM, New York, NY, USA, Article 6, 4 pages. DOI: https://doi.org/10.1145/2723147.2723155

[14] https://prototypefund.de/project/documents-for-democracy/
[15] https://github.com/svanteschubert/en16931-data-extractor
[16] https://prototypefund.de/project/papierloser-alltag/