PhUSE EU Connect 2018

Paper TT06

Breaking free from Microsoft Word

Dominik Habel, Bayer AG, Berlin, Germany

ABSTRACT

Microsoft® Word has dominated the document processing market for over 20 years and has become the go-to software in almost every office environment. After Word was first released, it quickly surpassed its competitors and became the best option available. However, the IT world experienced a drastic change since then affecting the very fundamentals of software. Web 2.0, open-source software, Big Data processing and interconnectivity are hugely important concepts of the modern world that Word struggles with. Microsoft's attempts to modernize Word as well as the sheer overuse of Word in offices brought up a variety of new problems that often remain unaddressed. This presentation will point out the biggest issues with Word from an IT perspective, discuss viable alternatives and show a real-life project trying to break free from Word when writing TFL Specifications.

INTRODUCTION

Microsoft Word is one of those programs that do not need a big introduction. It is by far the most widely used text processing software on the market with no end in sight. Taught in schools, required from job applicants by companies, it is one of the most important applications in every professional field. No matter where you look, you will doubtlessly see Word in use, albeit service, production, health, media, science, IT, administration or pharma.

Word's ascent started back in 1981 when Microsoft hired a couple of engineer's with GUI word processing experience from a competitor to begin developing a text processor on their own. Two years later, in 1983, Microsoft announced it as "Multi-Tool Word" which was renamed later on. Word was, and still is, a so called WYSIWYG word processor. That charming abbreviation stands for ***W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et* meaning the end result (e.g. after printing) will look like what you see on the screen while creating the document.

Multi-Tool Word had some significant advantages over its competitors. One of those being the fact that it was designed to be used with a mouse. Also it was able to use the graphics card on IBM-PCs to display bold and italic text passages whereas its competitors could only display plain text either with mark-ups or certain font colors. Those competitors (WordStar[1], WordPerfect[2] and IBM PC Text[3]) posed a problem since they were way more popular and inhibited Word's growth in the IBM PCs market segment. Luckily, Microsoft ported Word to Apple's Macintosh very early where it started to thrive and eventually became the market leader. In addition to that, the later on rise of popularity of Windows and constant improvements of Word itself contributed to the unique growth that led to a position where the market share is far beyond any competitor and the market-domination has been undisputed for many years.

Nevertheless, the IT world experienced drastic change since the 80s. The very fundamentals of how we develop software, how we distribute software and how we use software have changed. So can a concept that has been developed in the beginnings of Personal Computers hold up to the expectations set by modern IT? In the next sections various problems with Word will be listed, explained and discussed. Furthermore, you will find examples of software products that do not have these problems or do a better job of handling them.

PROBLEMS WITH WORD

PROPRIETARY SAVE FILE FORMAT

Word's save file formats are proprietary meaning they are a company secret and the exact implementation are only known to Microsoft. This was certainly true for the (now deprecated) format *".doc"* whose specifications were published way after it was retired and superseded by *".docx"*. Relying on proprietary data format can pose a problem due to the lack of competition and therefore alternative software. The risk of running into a vendor-lock-in increases leaving you at the mercy of a company to provide you support, interoperability and supplemental software.

With the XML based format *".docx"* or OOXML (Open Office XML) hopes in the IT community were high that Microsoft would abandon their proprietary approach since OOXML was presented as an open format that even got accepted as a standard by the International Organization for Standardization (ISO). Although this is true, people quickly discovered that Microsoft diverges from its own document standard and uses a non-standard implementation of OOXML. As a reason Microsoft stated compatibility reasons with older versions. That led to a multitude of different standards (OOXML 2007 Transitional, OOXML 2010 Transitional, OOXML 2013 Transitional) which create issues with interoperability with other non-Microsoft programs. In those you will hardly ever have the same result opening a

file as you have in Word directly. Also Microsoft uses proprietary fonts ("C-Fonts", e.g. Cambria, Calibri) as a default that other companies are not allowed to use, rendering them unable to properly open Word files that contain those fonts having to either license the fonts (shutting non-commercial software off) or resort to exchanging these fonts with standard ones which will certainly affect the look of the document.

There are several examples for non-proprietary file formats that are suitable for text documents. The most prominent and closest to OOXML would be OASIS Open Document Format for Office Applications (short: OpenDocument, ODF). ODF originated from the Apache OpenOffice software package[4] but became an ISO-approved international standard in 2005. Technically, it works very similar to OOXML but it is completely open-source and easier to implement than Microsoft's standard. Due to it being an ISO-approved format, ODF is supported by lots of programs other than OpenOffice. Some of the more famous ones are Calligra[5] and LibreOffice[6].
A different approach to text document formats are HTML based editors. With HTML being a very powerful, highly standardized and internationally accepted markup language, it can provide you a very reliable way of managing text documents. There are quite some WYSIWYG editors for it out there, e.g. TinyMCE[7], Froala[8] and CKEditor[9].
Some plain text formats could also serve as a great document format in some use cases. Namely JSON and XML are commonly used formats not only for their original purpose (browser-server communication) but also for documents. Although it can be difficult to use them in some use cases since they do not offer formatting options easily, their main advantages lie in a different field which will be covered in the next section.

BINARY SAVE FILE FORMAT

While the old *".doc"* format undoubtedly is binary, opinions clash with OOXML files. This may be caused by the definition of binary files that leaves room for interpretation or the way that *".docx"* files work. The Linux Information Project defines binary files as the following:

"A binary file is any file that contains at least some data that consists of sequences of bits that do not represent plain text."[10]

The quoted website is highly recommended by the author because it provides some more interesting information regarding binary files and even touches some of the topics in this paper as well. Going by this definition, testing a file on binarity is easy. You try to open it in a simple text editor. If you can read the content, it is non-binary. If it looks like gibberish, it is most likely a binary file. Now, by that logic OOXML files are clearly binary. The reason why there is room for discussion is because *".docx"* files are ZIP compressed files. You can open any OOXML with your preferred ZIP handling software (e.g. 7Zip[11], WinRAR[12]) which will reveal a folder structure with XML files building the OOXML file (ODF files are build similarly).

Plain text files have a multitude of advantages over binary files. One being, that you can open and read them independently from the operating system, installed software and time. In 30 years OOXML might not be in use anymore but you'll always be able to read plain text. More importantly, a lot of modern IT architecture depends on plain-text files. Using binary formats will lock you out of most version control systems, repository setups, databases and automation systems. GIT[13] being a great example of a software having standard software status in the IT world but remaining widely unused in the pharma sector. Using GIT can be very useful, not only for handling program code but also for documents. It offers powerful tools for versioning, change tracking, user management and collaborative work. Furthermore, Continuous Delivery and Continuous Integration pipelines often integrate GIT and rely on the presence of plain text files as well, a topic that Andrew Karpow covered in 2017's PhUSE[14].

Standards and documents planned to be resources for software are almost necessary to be in a plain-text format like JSON or XML to allow easy import, parsing and versatility. Whereas using information from manually created Word documents requires more effort and has a higher susceptibility to error that is in Word's case amplified by the unsteadiness and uncertainty of the exact specification of the file format.

LARGE FILES

Word struggles with large files which can particularly be noticed with TFL documents. Those can contain hundreds of pages packed with fields, page-spanning tables and images that have to be calculated and rendered when opened. Even with up-to-date hardware, Word will slow down considerably, often to the point where it will stop responding for several minutes only to crash shortly after. This can result in a massive problem not only for TFL documents but also other large Word files when you need to "quickly" check or edit something. In a world, where big data is more and more prominent and document sizes increase, this will be a problem more often than not in the future.

Other document concepts are able to handle this better. Mark-up languages allow the user to separate content creation and processing-heavy rendering, some application only provides content that is currently needed limiting the amount of data to process, some document concepts are simply more light-weight compared to Word increasing file handling speed.

NON-SPECIALIZED

Word is your all-rounder text processor. You can use it for your notes, letters, program specifications, book writing, invoices, invitations and so on. All that comes with a price though. The software is overloaded with functionalities that are packed inside one nested menu after another to suit all users. Many features were added on the way, stacked into the application. Some features are really high-level, some are more technical. Usually, only a tiny fraction of these functionalities is used and even needed. Also it can cause confusions on which features should be used for which task. Often there is no clear "right" way of doing something.

There is a lot of specialized software on the market which may be able to increase the productivity and quality of your work since it can offer features tailored to your needs without all the overhead. For quite some use cases it can be very helpful to take a look at different, more fitting approaches. For your notes, there are apps like SimpleNote[15], book writers may prefer Scrivener[16] or WriteRoom[17], for brainstorming LucidChart[18] or Bubbl.us[19]. It is definitely worth looking around for other software with different concepts.

SOME POSITIVES ABOUT WORD

Despite all the problems mentioned above there is a reason why Word is leading the text processing market. The WYSIWYG paradigm still works after all these years. Users can see what they're doing, they do not need any IT knowledge to create the required output. In addition, Word is accessible in most companies and even on private PCs, it can be considered as a standard software which you can expect people to be able to use. With Word, you don't really need to worry if your customers can open it because it is almost implicitly expected from them. Also, Word just works. Of course, it has its problems, it's slowish, overloaded and buggy at some points but you will be able to create what you need in a considerable time without missing functionalities. In the recent past, many great improvements found their way into the software. The styles help a lot, the formula editor became more intuitive and the possibility of programming small VBA macros can be a nice tool as well. With Office 365 and Word Online, Microsoft did the first step of moving into a cloud environment. Hopefully, they will further pursue this path and improve upon Word Online, so that it feels more like a thought through concept rather than a quick port into a cloud environment.

BREAKING FREE FROM WORD

The intent of this paper is not to convince readers to abandon Word. It is way too embedded in the professional and even private office environment, so that it would be problematic to get rid of it. Additionally, Word can be the correct choice in a lot of use cases, where the advantages outweigh the problems. Nevertheless, it seems that especially larger companies simply choose Word for everything forgetting that sometimes there are better alternatives available. Breaking free from Word means starting to question the use of Word, it means to reconsider its use when it makes sense to the department/company and to open up for other software on the market. Unfortunately, there is no blueprint on where to replace Word, it has to be discussed and evaluated on an individual basis.
When looking at startups, a big factor of their innovation power and fastness certainly is their flexibility in software they use and that they go through this selection process. They are not bound to legacy software or license deals. Instead they can search the market for the best option available and what works in conjunction with each other which can lead to a significant increase in speed, quality and efficiency that large companies are in danger to lose.
In the following section, a few alternatives that have the potential to supersede Word in bigger use cases will be described. Additionally, such a project from the experience of the author will be presented shortly.

GOOGLE DOCS

Google seems to be the only real competition to Microsoft's whole Office package. Usually when people talk about Microsoft Office alternatives, they either come up with other office suites or individual solutions for every Office product. Both ways have indisputable negatives. Alternative office suites like LibreOffice (which was created out of OpenOffice which is not active anymore) often come up. Although they do not share all of Microsoft Office's weaknesses, they have their own. They fix some bugs and issues users complain about with Microsoft's product but introduce new inconveniences and bugs. LibreOffice may be interesting for companies that value open source projects. It is free, comes for every platform, even Unix systems, and is backed by the Document Foundation. So it is still worth considering for either new companies building their IT or for companies who rebuild their IT landscape on a bigger scale. Convincing users to use something that feels to them like a Word clone probably is a lost enterprise from the start. Moreover it is also not a good idea to replace Office by increasing the amount of applications. Naturally, you could find 3 programs that can in conjunction cover all your Word use cases but you will bring a lot of new complexity into your organization as well as tons of training effort, support cost and maintenance issues.
Google with their G Suite[20] offers a solution that is neither a clone nor an island solution while additionally following recent trends in software distribution. G Suite does not only offer the counterparts for Word, Excel and PowerPoint but also includes mail, communication, a calendar, social networking, website creation and survey applications. The difference to Microsoft's product palette is that Google's services are cloud based with all the benefits that implies. The first one being they're available from anywhere, from any browser without having to download or install a single thing. Furthermore Google does all the file management and system architecture work for you. The files are auto-saved almost instantly when changed, they're backed-up automatically and stored on Google's servers. So you

simply access them and leave the rest to the cloud service. Google also took the complexity out of their applications. Docs for example only offers a very small range of functionalities that are sufficient for most documents you need but Word power users will miss almost all of the more advanced features. Therefore Google Docs provides the option to include Add-Ons from 3rd party vendors which can add new functionality as it is needed. Moreover, G Suite's focus lies much more on collaboration. It is very simple to set access rights for other people, multiple people can work on the same document at the same time while others can see where their colleagues are and what they're doing. You can start discussions in the comments section or in the real time chat and there is a change history available where you can again track all the changes made. Exporting the documents is available for a multitude of file formats including OOXML.

It is highly recommended by the author to take a look at the G Suite's apps especially Google Docs when the focus of your work in on collaboration, availability and usability based on personal experiences in working on documents in groups of up to 20 people (often editing with multiple people simultaneously).

LATEX

LaTeX[21] is a very controversial software. It can almost exclusively be found in the science community used for publications or university papers. Outside of that, it is very rarely used, in companies it has very little meaning at all and even in universities it slowly loses its status. Core of the problem being that it is more a programming language than a word processor repelling people who never came in contact with markup languages. Further LaTeX has a steep learning curve and is not as intuitive as Word which further scares users away. Yet it comes with various benefits that done in the right way have the potential to be more efficient than using Word and of better quality.\par

LaTeX is a free software using plain text files to compile mainly PDF files but also other formats are supported like RTF and HTML. Its markup language nature enables the use of variables, very precise formatting and equal output on different systems. Just like other programming languages, running it on other architectures will produce the exact same result whereas Word documents can be different on different PCs let alone operating systems. Additionally, there are lots of extensions available that can add valuable functionalities.

Still, the complexity and with that the opportunity for error of LaTeX is way higher than Word's but remedy can be provided. First, there are very helpful editors available, TeXstudio[22] being a quite advanced one. Those can help tremendously with overview, markup assistance, output preview, log and error messages and much more. The even bigger opportunity is the fact that content can be separated from document setup. That means the difficult part requiring more thorough knowledge of LaTeX (setting margins, building title page, defining variables, setting the look of the headlines, footnotes, page header and footer) can be outsourced to specialized teams and the content creators can suffice with very few commands. The basic content of this paper has been done with only a couple LaTeX commands (mainly: Starting a new chapter ("\chapter{chapter_name}"), starting a new section ("\section{section_name}") and ending a text paragraph ("\par")) while TeXstudio offered code completion and assistance via the user interface (formatting buttons, symbol choice menus).

To sum up, it would be recommended to look into LaTeX, if you create documents which base structure is consistent between documents of the same type and do not change in every situation. Therefore you can create LaTeX templates that you then distribute to the content creators that can work with it given that the users have some basic understanding of programming principles which, in a statistical programming environment, is a valid assumption. That alone can help to improve document quality by a lot but the largest potential is when embedding it into a version controlling, maybe even Continuous Delivery system which then can improve the speed of document creation as well while simultaneously opening up options to automate certain steps in the process.

TALIFI

TaLiFi is the name of a project of the author aiming at retiring Word in the TFL Specification creation process. Up until now the creation consisted purely of plain manual work in Word. The layout of the document was defined in a template and the content was partly handcrafted, partly taken from several standard table catalogs where frequently used shell tables and listings ("XX" instead of values) are collected. This approach was lacking both speed and quality. Assembling shell tables for the document consisted on "painting" them from scratch or searching for the appropriate ones in the catalogs that consist of dozens of elements themselves creating a very unfriendly and unappreciated experience for the creators. The quality of the outcome also suffered from that. The specifications were a collection of references to unlinked catalogs mixed with drawn tables accompanied by notes regarding changes in the title, footnotes, even the table layout itself. This resulted in hard to review documents which then poses the risk of missing mistakes and possible problems in later phases of the study.

It was assessed that a system for creation TFL specification bears enormous opportunities. Some of those being better quality, machine-readable catalogs that can be easier maintained, user-friendlier work-flows and improved standardized and machine-readable outputs that are easier to understand, review and adjust. Furthermore such a system opens up the possibility of implementing more advanced features like automation of certain work-flows, having a company-wide shell table/listing repository, connecting to the SAS environment (inserting footnotes, titles or even starting the creation of datasets or table programs) and much more.

TaLiFi started as a Java application with an user interface that enables TFL specification creation by providing means to collect study-specific as well as catalog tables, listing and figures in a specifically for this purpose developed interface while automating the creation of the final Word document output. The concept includes the ability to browse standard catalogs, add elements to the specification by simple drag and drop gestures, enabling

certain adjustments like setting the population or removing optional columns, adding external images and tables by simply dragging them into the application and further usability improvements. For that purpose, the Word based standard catalogs were not usable, so they were converted into JSON format allowing them to provide the necessary data to the application while the user interface handles the display of the elements as well as their maintenance. Save files from TaLiFi remain in JSON format and only for the output, they are converted into a Word document satisfying layout requirements. This ensures that subsequent processes will not be negatively affected by this new system.

CONCLUSION

In summary it can be said, therefore, that there is no simplistic way of replacing Microsoft Word. It way too embedded into the IT landscapes of companies and for a lot of use cases the pros of keeping Word outweigh the advantages of introducing other software. Yet it is necessary to critically assess Word's impact in areas where you want to use the information currently stored in OOXML documents in a software to avoid Word blocking your access to data science and automation. Besides that, the market place for software solutions is flourishing with new ideas and concepts that are worth exploring and that may have the potential to benefit your company in their work.

REFERENCES

[1] http://www.wordstar.org/
[2] https://www.wordperfect.com/en/
[3] https://de.wikipedia.org/wiki/PCText
[4] https://www.openoffice.org/de/
[5] https://www.calligra.org/
[6] https://de.libreoffice.org/
[7] https://www.tiny.cloud/
[8] https://www.froala.com/wysiwyg-editor
[9] https://ckeditor.com/
[10] http://www.linfo.org/binary_file.html
[11] http://www.7-zip.de/
[12] https://www.winrar.de/
[13] https://git-scm.com/
[14] Catching Up: Continuous Integration Pipelines For Clinical Analysis
[15] https://simplenote.com/
[16] https://www.literatureandlatte.com/scrivener/overview
[17] http://www.hogbaysoftware.com/products/writeroom
[18] https://www.lucidchart.com/
[19] https://bubbl.us/
[20] https://gsuite.google.com/
[21] https://www.latex-project.org/
[22] https://www.texstudio.org/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Email (private): dominik-habel@web.de
Email (work): dominik.habel@bayer.com