LibreOffice Writer
für pdf technical manuals

User stories - de

Dr. Hartmut Schorrig
www.vishia.org

2024-01-29

Table of Contents

Figures

1 Zweck des Dokumentes

Der Zweck ist eigentlich eine User-Story mit dem Hintergrund, Verbesserungshinweise für LibreOffice zu geben.

LibreOffice steht im Wettbewerb mit Microsoft Word, aber auch mit OpenOffice. Es kann nicht pauschalisiert gesagt werden – MS-Word wäre besser, professionell, gar nicht teuer, also bitte was soll das. Nein, MS-Word ist nicht besser. Es ist grundsätzlich sogar eine politische Entscheidung (politisch = für die Menschengemeinschaft), eine Software zu nutzen, deren innere Codierung zugänglich ist, als Plädoyer für Open-Source-Software, nicht weil es kostenlos ist, sondern frei zugänglich.

Also lohnt es sich, vollständig auf LibreOffice zu setzen. Die Kompatibilität mit MS-Word, das weit verbreitet ist, ist gegeben. Zu beachten ist Abgrenzung und Kenntnis zu OpenOffice, was ebenfalls Open Source ist. Der Unterschied oder die Entscheidung zwischen beiden ist hier eher „Vielseitigkeit tut gut“. Wir haben also zwei Tools, die weitgehend kompatibel sind, vielleicht etwas in Details unterscheiden, aber auch von verschiedenen Communities gepflegt werden. Man kann sich im Wesentlichen für eines der beiden entscheiden (beispielsweise wenn es um Wartungsverträge bei kommerzieller Nutzung geht), aber doch das andere auch nutzen.

Meine Intension ist, für LibreOffice beizutragen, als Teil dieser Community, also auch im wesentlichen zu nutzen, aber doch einen Seitenblick auf OpenOffice zu haben. Von MS-Word habe ich Abstand genommen, seit es diese unübersichtlichen horizontal angeordneten Styles anbietet und die Menüleiste (SAA-Menü) abgeschafft hat.

Das Dokument beschreibt meine Herangehensweise und deren Erfahrungen, als Impuls auch für andere Nutzer. Es geht um Usability-Stories (Nutzbarkeitsgeschichten).

Englisch oder Deutsch?

Häufig werden technische Dokumente in Englisch geschrieben. Das ist sinnvoll, um sie nicht zweimal pflegen zu müssen, oder automatisch schlecht zu übersetzen und bei Änderungen immer nach übersetzen ohne redigieren wenn erwartbar ist das Leser aus verschiedenen Teilen der Welt kommen.

Dieses Dokument ist in deutsch, weil es sich zunächst an eine deutsche Community wendet, aber natürlich ist das in diesen Kreisen übliche Wort Community nicht eingedeutscht, wie auch weitere bekannte Anglismen.

Allerdings ist es meine Erfahrung, bei Tools die Bedienung auf Englisch einzustellen, da die deutsche Bedienversionen doch oft mit einem unbekannten Sprachschatz operieren, den man nicht so kennt wenn andere Tools nur in Englisch bedient werden.

Daher sind in diesem Dokument Libre-Office bezogene Bedienhandlungen in Englisch angegeben.

2 Erscheinungsbild der Dokumentation

2.1 Technische Dokumenation, Doppelseite im pdf

Für Technische Dokumentationen sind entweder HTML-Seiten üblich, insbesondere im Internet, aber auch pdf, letzteres ebenfalls auch im Internet. Offizielle weitergebbare Dokus sind eigentlich immer pdf. Man braucht ggf. beides.

Pdf hat vorrangig den Vorteil, dass es in sauberer Form ***weitergebbar*** ist, auch ausgedruckt als Anleitung liegend auf dem Tisch benutzt während der Hard- und Softwareentwicklung.

Doppelseitenansicht im pdf: Ein weiterer Vorteil ist die Seitenbindung. Um gut zu erklären, sollte ein Thema möglichst auf einer Doppelseite dargestellt werden. Die Doppelseite ist auf einem etwas größeren Monitor gut lesbar darstellbar, und zwar nicht als „fortlaufend“ sondern Seiten schaltend. Vorteil ist hierbei, der Blick kann fokussiert werden, beispielsweise auf Haupt-Überschriften links oben oder bestimmte Bilder, die beim Durchblättern am Bildschirm immer an der gleichen Stelle erscheinen. Wenn man etwas bestimmtes sucht, aber nicht genau weiß was, sondern nur eine Erinnerung hat „sah so aus...“, dann ist dies hilfreich. Freilich kann auch schnell nach einem Stichwort gesucht werden, aber auch da hilft die gleichbleibende Fokussierung auf die gesamte Doppelseite.

Dieser fokussierte Blick und die komprimierte übersichtliche Darstellung auf der Doppelseite ist der Hauptvorteil gegenüber html, dass weder die Seitenbindung hat noch die Information beispielsweise in Spalten flächig auf der Bildschirmfläche verteilt. HTML ist in dieser Hinsicht stehengeblieben bei der Auflösung der alten Monitore mit 1024 oder 1280 Pixel breite, die sowieso nur eine Seitenbreite darstellen konnten.

2.2 Zweispaltige Seiten oder Halbseiten

Zweispaltige Seite: Das Schreiben in Spalten ist bei Zeitschriften üblich, also eigentlich bei technischen Aussagen. Worin liegt der Vorteil:

Das Auge kann in einer Zeilenbreite von etwa 40 Zeichen Wörter und Wortgruppen als Ganzes erkennen. Man kann sozusagen senkrecht von oben nach unten lesen, um einen Text schnell zu erfassen, oder auch nur einen bestimmten Abschnitt wiederzufinden. Läuft die Zeile breiter, dann funktioniert dies nicht mehr. Das Auge muss eine rechts-links-Bewegung machen um zu erfassen und zu suchen. Dieses Wissen über die Sinnhaftigkeit des mehrspaltigen Schreibens ist in Zeiten von HTML offensichtlich verlorengegangen, eben weil ursprünglich die Bildschirmbreite sowieso begrenzt war. Auch bei Word & co ist es ein extra Aufwand. Den Vorteil sollte man aber bedenken.

Bilder passen oft in diese Spalten. Eine andere Variante, kleinere Bilder einzubinden, ist die Anordnung rechtsbündig mit links vorbeilaufenden Text (oft in der Wikipedia verwendet, als HTML-Stil gut verfügbar). Das ist bei zweispaltiger Schreibweise freilich so nicht mehr sinnvoll.

Übersichtsbilder sollten dann aber die gesamte Seitenbreite einnehmen. Es ist wichtig, wo sich diese Bilder im Seitenlayout befinden. Das sollte bei der Dokumentenerstellung bewusst beachtet und bestimmt werden.

Die Spalten sollten dann bis zum großen Bild links und rechts verlaufen, und unter dem Bild neu mit links und rechts beginnen. Das ist anders als die Gepflogenheiten in Fachzeitschriften, bei denen die Spalten

grundsätzlich auf die Gesamtseite bezogen von oben nach unten durchlaufen, das Bild unterbricht hier quasi nur flächig die Spalte als solche. Der Vorteil des Neuansatzes der Spalten unter dem Bild ist, das auch bei kleineren Monitoren (Smartphone) die zugehörigen Informationen links und rechts übersichtlicher dargestellt werden. Bei einer Zeitschrift hat man gesamte Seite etwa im Format A4 vor sich, sieht also die Spalte von oben bis unten als Ganzes, nicht unbedingt jedoch so am Bildschirm.

Gleiches für den Neuansatz der Spalten gilt auch für Hauptüberschriften, die dann auch über die gesamte Seitenbreite gehen, und damit eine deutliche Gliederung in der Dokumentenfläche bewirken.

Lesbarkeit der Spalten am Mobile („Handy“): HTML hat den Vorteil dass es je nach Vergrößerung (Zoom) neu rendert, der Text läuft immer auf die Monitorbreite bezogen. Das ist bei pdf grundsätzlich nicht so, es wird eben nicht neu gerendert sondern die Seitenaufteilung ist fest, mit den oben beschriebenen Vorteilen. Aber daher muss beachtet werden, dass das Ganze gut lesbar ist.

Nun ist eine Spaltenbreite von ca. 40 Zeichen, wie sie in diesem Text verwendet wird, bei den gängigen Monitorgrößen von Mobiles im Hochformat noch gut lesbar. Man bekommt dann die halbe Druckseite etwa in der Höhe gut sichtbar, und kann schnell auf die Nachbarspalte daneben rüberschieben um weiterzulesen. Wenn in der Regel die Seite in der Mitte durch ein Bild in Seitenbreite, eine Zwischenüberschrift oder auch nur eine Trennlinie unterbrochen ist, also zwei zueinandergehörende Spalten eigentlich nur etwa eine halbe Seite lang sind, dann ist dies fast ideal. Gegenüber HTML ist der Vorteil, man weiß wo was steht (fürs Zurückblättern und suchen), hat eine bessere Orientierung anstatt dem länglichen Scrollen bei HTML-Seiten.

2.3 Unterstützung des zweispaltigen Schreibens in LibreOffice

Es wird unterstützt, wenn auch nur *„tricky“*. Ein Abschnitt mit zwei Spalten ist eine andere *„Section“* als solche sichtbar im *„Navigator“* (als *Sidebar* auffindbar). Man kann leider diese Section nicht unterbrechen um zwischendurch wieder auf volle Schreibbreite zu gehen. Wählt man in der zweispaltigen Section wieder etwas „spaltig“ (Format – Columns) an, dann bekommt man die Spalte zweispaltig, was eigentlich nonsense ist. Es wird dann eine neue geschachtelte Section angelegt. Hier wäre das Handling, die Bedienerführung im LibreOffice also verbesserbar Es muss die Möglichkeit geben, inmitten der Section die Section aufzubrechen und auf drei Sections zu verteilen, oberhalb und unterhalb der Aufbrechstelle (Cursorposition) als zwei weiterhin zweispaltige aber nun getrennte Sections, und dazwischen eine Standard-Section, also über die gesamte Seitenbreite. Dann könnte man einfach ein über die gesamte Seite gehendes Bild oder eine Zwischenüberschrift einfügen.

Wie geht es im Moment (Version 7.5): Man geht an das Ende der Section, ab der dann wieder volle Seitenbreite gilt, fügt dort mindestens ein paar Leerzeilen ein, ordnen dann die unteren Leerzeilen wieder zweispaltig (markieren, Format-Columns, *„Current Selection“* anwählen, hat also eine neue Section zweispaltig und darüber eine Section mit der Seitenbreite. Dann lösche man den Text, der darunter stehen soll, aus der zweispaltigen Section darüber in die Zwischenablage (*ctrl-X* oder *sh-Del*) und füge diesen mit *ctrl-V* oder *sh-Insert* in die untere Section ein. Etwas Handling notwendig aber man bekommt damit was man will.

Tipp: *„View – Text Boundaries“* einschalten.“

Es wäre gut, wenn die Spaltenformatierung als Style zuordenbar ist (indirekte

Formatierung der Spaltenansicht). Ich habe dafür keine Möglichkeit gefunden. Es gehört nicht zur Page-Formatierung, das ist korrekt. Es sind Sections innerhalb der Pages, also etwas zwischen Paragraph und Page angeordnet. Beispielsweise ist in diesem Dokument der Abstand zwischen den Spalten auch mal etwas unterschiedlich, da ich zwischendurch die Maßeinheit für Texte von cm auf Pt geändert habe.

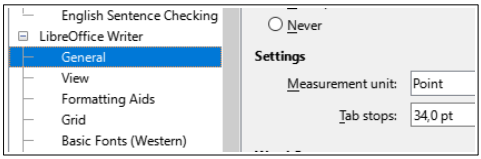

Figure1: LibreO/MeasurementUnitText.png

2.4 Schrift und Seitengröße

Folgendes hat sich als Erfahrung etabliert:

Kein A4-Format verwenden, sondern A5.

Schriftgröße 9 pt für Normalschrift.

Damit bekommt man zweispaltig etwa 40 Zeichen in der Spaltenbreite, gut lesbar.

Druckt man dieses Dokument als „Broschürendruck“, von manchen Druckertreibern direkt ohne Zusatzaufwand unterstützt, so wird dies ohne Verkleinern auf Doppelseiten A5 passend ausgedruckt mit Vorder- und Rückseite richtig sortiert. A4 ist das meist verwendete Papierformat in Druckern, passt zur handlichen A5-Broschüre zusammengefaltet für den täglichen Gebrauch oder als Ausdruck praktisch weitergebbar.

Die andere Variante wäre A4-Schreiben, 12 pt Schrift ergibt etwa auch 40 Zeichen pro Spalte, und Ausdruck dann mit Verkleinerung A4 auf A5. Das ist aber komplizierter handelbar.

Ein weiterer Vorteil des A5-Formats: Es entspricht in der Doppelseite der Größe eines Normal-kleinen Monitors (14..15“). Ich arbeite hier auf einem Notebook Monitor mit 13“, geht auch gut. Man kann beim Schreiben des Dokumentes also auch auf die Doppelseitenansicht gehen, sieht was wohin kommt, hat sozusagen einen Wysiwyg-Blick (*„What you see is what you get“*). Von Libre office wird das gut unterstützt mit der Einstellung *„View-Zoom View Layout Columns:2“* wobei hier die Bedeutung von *Columns* nicht mit dem zweispaltigen Schreiben zu tun hat sondern mit 2 Seiten nebeneinander, dann jeweils mit 2 Spalten, man sieht als 4 Spalten. Das ist in der Bezeichnung etwas irritierend. Im Menü müsste besser statt *„Columns“* *„Pages“* stehen, das entspricht dem Fakt.

2.5 Book-Mode Ansicht ?

Es gibt eine wichtige Regel für Bücher, die bei Office-Tools und pdf-Viewern offensichtlich immer mal in Vergessenheit gerät: ***Die ungerade Seite ist rechts.***

Dies ist nun extrem wichtig, wenn man in Doppelseitenansicht gut präsentieren will. Es gibt zwar bei Büchern die Regel, ein Hauptkapitel solle auf der rechten (ungeraden) Seite anfangen. Doch diese Regel ist nun gerade ungünstig da man auf zwei nebeneinanderliegenden Seiten das Kapitel überschauen können sollte. Also, das Nebenkapitel bitte immer auf einer geraden Seite beginnen um es auf zwei Seiten, links gerade, rechts ungerade, zu sehen.

Nun könnte am sich in Beliebigkeit begeben, denn insbesondere die Browser können hier alle möglichen Ansichten. Aber mit der Beliebigkeit kommt man an Grenzen, wenn die Doku dann doch einmal gedruckt gewünscht ist. Beliebigkeit sollte Standard weichen, und der sagt nunmal, ungerade Seite ist rechts. Somit wird es einheitlich (der Standard gilt sogar bei amerikanischen Büchern).

Nun ist es leider so, dass zwar die meisten pdf-Reader die Anwahl *„Seitenlayout – Titelseite bei doppelseitiger Ansicht“* unterstützen, im Klartext heißt das *„ungerade Seite rechts“*, aber leider tun das nicht alle Viewer und insbesonder nicht alle Browser für pdf-Ansicht.

Folglich muss man tricksen. Das Gesamt-Dokument beginnt mit einer leeren Seite, und dann mit der Titelseite, deren Seitennummer über einen manuell eingerichtete neuen Page-Style ab hier mit 1 beginnt. Dann hat man die ungerade Seite rechts, und stellt im pdf-Betrachter nicht den „Sondermodus Titelseite“ ein, sondern den sogenannten Normalmodus, der unstandardiserterweise mit der ersten Seite links beginnt, aber die zweite Seite rechts trägt dann die Nummer 1.

So geht man um mit etablierten Standards aus den Urzeiten von Gutenbergs Buchdruck, die aber den Browser- und pdf-Softwareentwicklern offensichtlich nicht so genau bekannt sind.

2.6 Ordnen der Dokumentation auf den Seiten

Bei Latex wird oft beklagt, dass man nicht weiß wo das Bild hinkommt. Es schnippst beim Rendern irgendwohin, und die Ausrede ist dann, es würde ja über den Querverweis der Bildnummer gefunden, auch wenn es auf einer anderen Seite steht.

Nun ist aber für eine gut leserliche Dokumentation es sehr wichtig, dass alles richtig beieinander ist, dass man nicht blättern und suchen muss um zu verstehen.

Daher sollte ein ganz anderes Prinzip gelten:

Die Seitenaufteilung ist Teil des Schreibens der guten Dokumentation.

Eben daher ist es von Vorteil, direkt in LibreOffice mit Beachtung der Seiteneinteilung zu schreiben, bei Erzeugung eines pdf genau zu wissen, dass die Seiten genau so aussehen werden (*What you see is what you have*). Man darf also ein wenig Mühe investieren, damit ein Bild dort steht, wo es gut in den Textfluss passt. Man darf etwas Mühe aufwenden, damit ein Thema genau auf die Doppelseite passt. Man kann eine nebensächliche Erklärung dann ein wenig kürzen, weglassen, woanders hin schieben und hier nur einen Verweis anbringen, je nachdem.

> *Ich habe mich selbst über Manuals eines bekannten US-Chipherstellers geärgert, bei dem wichtige Tabellen mit Bitbedeutungen verschiedener Register je nachdem zufällig irgendwo auf der Seite beginnen, mal oben, unten, links oder rechts. Sucht man ein bestimmtes Ctrl-Register, dann hilft beim schnellen durchblättern, wenn dessen Bezeichnung immer oben auf der Seite steht und die Bits darunter. Es sollten alle 16 auf die Seite passen und nicht wegen nebensächlich wiederholender Erklärung die Tabelle über 1 ½ Seite gehen, so dass das nächste Register am Bild unten beginnt.*

Ein Hauptkapitel einer umfangreichen Dokumentation darf wie bei Büchern üblich rechts auf der ungeraden Seite beginnen, die linke Seite soll dann wirklich frei sein, und sollte auf dieser einen Seite den Überblick darstellen. So dass die zugehörigen Unterkapitel dann links auf geraden Seiten über eine Doppelseite sichtbar präsentiert werden.

Eine leere Seite, damit das nächste umfangreiche Unterkapitel zweite Stufe links auf der Doppelseite beginnt, ist durchaus nicht störend.

In dieser Dokumentation sind die Kapitel Level1 so kurz, dass die Regel nicht so wichtig ist.

3 Formatvorlagen (Styles)

Formatvorlagen gibt es schon seit den ersten „Word für DOS“ Versionen und sicherlich auch im damaligen Star-Office. Für professionelle Dokumentationen ist diese Herangehensweise schon vor der Zeit der computerbasierenden Dokumentationen üblich. Jedoch wird seit je her einen Parallelität der direkten Formatierung und der indirekten Formatierung mit Styles angeboten, der Anwender kann nach Belieben, seinem Gefühl und Kenntnisstand wählen, auch innerhalb eines Dokumentes. Latex ist konsequenter, es schreibt die Verwendung von Styles zur Auszeichnung vor.

Die zweite Verwaschung der Handhabung ist damit gegeben, dass auch kommerziell standardgemäß angebotene Formatvorlagen den Namen nach dem Aussehen wie beispielsweise *„blue1“* bis *„blue3“* erhalten. Der Fehler hierbei ist, die Formatvorlagen sollen nach dem Zweck benannt sein, also beispielsweise *„Quotation“* und nicht nach dem Aussehen wie *„italic“*. Das Aussehen kann dann für den jeweiligen Zweck passend eingestellt werden, wobei für Zitate es schon üblich ist, diese kursiv zu setzen, aber es ist keine feste Regel.

Nun verwende ich in meinen Dokumentationen nur die indirekte Formatierung. Das ist konsequent. Jedwede Mischung mit direkter Formatierung ist Murks. Die beste Tastenkombination ist *ctrl-M „Format - Clear direct formatting“*. Nun ist es leider so, dass üblicherweise *ctrl-I* die direkte Formatierung *„italic“* einschaltet, ein schneller Griff. Um die indirekte Zeichenformatierung *„Quotation“* anzuwählen, sind schon einige mehr Mausklicks nötig. Mit der Tastatur geht es nicht. Es wäre gut, bestimmte, insbesondere Zeichenformatierungen auf Hot-Keys legen zu können. Etwas in diesem Zusammenhang: Es wäre gut, wenn eben neu angelegte Styles unter „Applied Styles“ angezeigt werden würden, da diese ja zum Anwenden eben erzeugt wurden. Es ist lästig, zunächst wieder „All Styles“ oder „Custom Styles“ anwählen zu müssen, um den eben angelegten Style wieder zu finden.

Die konsequente Verwendung nur der indirekten Formatierung könnte besser unterstützt werden, in dem unter „Tools - Customize“ alle Bedienmöglichkeiten der direkten Formatierung abgeschaltet werden können (wenn dies der Wunsch des Bedieners ist).

Der Sinn und die Verwendung der indirekten Formatierung muss man allerdings dem unbedarften Nutzer gut erklären. Für Otto-Normal-Textschreiber muss es sicherlich auch die direkte Formatierung geben.

4 Gliederung, Inhaltsverzeichnis

Grundsätzlich ist eine gute Gliederung eines Dokumentes wichtig für eine gute Auffindbarkeit von Themen. Das wird von Office-Tools von je her gut unterstützt, muss aber ggf. dem unbedarften Nutzer ebenfalls erklärt werden.

Auch die Erstellung des Inhaltsverzeichnisses funktioniert gut (*„Insert – Table of contents ...“*).

Mir ist es aber nicht gelungen eine automatische Nummerierung anzuwählen. In älteren Versionen, auch von MS-Word ging das sehr einfach. Seit einer bestimmten Version von Libre Office ist dies offensichtich irgendwie in *„Tools – Line Numbering“* versteckt. Mir ist es nicht mehr gelungen, die automatische Nummerierung für Header-Styles zu aktivieren. Folglich verwalte und korrigiere ich die Nummern manuell – etwas arbeitsaufwändig bei Änderungen der Gliederung, geht aber irgendwie nicht anders. Die Nummerierung müsste an sich eine Eigenschaft des Absatzformates sein. Heading1.. ist ein Absatzformat, dort müsste eingestellt werden ob eine automatische Nummerierung erfolgt. Der Startpunkt der Nummerierung ab 1 sollte immer ein übergeordnetes Nummeriertes Absatzformat sein, möglicherweise auch ein Quasi-Dummy-Format (ohne Textinhalt) wenn zwischendurch eine Nummerierung wieder ab 1 anfangen soll. Für den Anwender dürfte dies leichter verständlich sein als irgendwelche Spezialeinstellungen, es ist dann im Navigator entsprechend auch gut ausweisbar.

Es gibt in Libre-Office einen störenden Effekt, der sich insbesondere für die Heading-Styles bei manuellen Seitenumbrüchen immer wieder zeigt:

Bei MS-Word war mindestens früher das Absatzformat mit dem anzeigbaren Absatzendezeichen (das pi) verbunden. Wenn man das Absatzendezeichen kopiert, also ab vor dem pi bis hinter den Umbruch dieses eine Zeichen in die Zwischenablage hinein tat, dann war das Absatzformat in der Zwischenablagen. Das war nicht unlogisch, man war es gewöhnt.

Das funktioniert aber nicht (mehr) bei LibreOffice. Wo das Absatzformat verankert ist beim Kopieren, hat sich mir noch nicht erschlossen.

Wichtig ist folgende Regel: „What you see is what you have“. Also nicht „get“ sondern *„have“*. Eine Regel, die überall gelten sollte. Auf einem 5 € Schein steht eine 5 und Euro in mehreren Sprachen, *„was du siehst das hast du“*. Zusätzlich kann man selbst kontrollieren, ob das Wasserzeichen in Ordnung ist. Bei einem Auto sollte man sich darauf verlassen, dass die Tank-Füllstandsanzeige stimmt. Beim Trabant gab es für diesen Zweck einen manuell einzutauchenden Mess-Stab, da wusste man genau wieviel noch drin ist. Bei neuartigen akkubetriebenen Fahrzeugen wird es schwieriger, kann auch schonmal falsch sein.

Die Regel auf LibreOffice angewendet muss also sein: Feldfunktionen müssen einschaltbar sein oder in der Statusleiste angezeigt wenn der Cursor dort steht. Es muss klar sein, wo die Information zwecks Kopieren verankert ist.

5 Arbeit direkt in der XML-Codierung

Das odg Format ist ein zipFormat. Untersuchen ist möglich durch Kopieren einer odt-Datei mit der Extension .zip, dann auspacken.

Die dort enthaltene content.xml enthält den textuellen Inhalt. Styles.xml sind die allgemeinen Formatvorlagen. Siehe:

https://help.libreoffice.org/6.2/de/text/shared/00/00000021.html Überschrift: XML-Dateiformat

Zitat von dort:

Die Verwendung von Einzügen und Zeilenumbrüche kann in den Experteneinstellungen durch das Ändern des Wertes der Eigenschaft /org.openoffice.Office.Common/Save/Document PrettyPrinting auf true aktiviert werden.

Das ist die Voraussetzung, mit den XML-Daten arbeiten zu können. Die Einstellung kann generell vorgenommen werden. Nachteil ist evtl. eine minimal größere Rechenzeit oder minimal größerer Speicherplatz. Daher kann man verstehen, dass das XML-Format original so dicht wie möglich gepackt wird.

Grundsätzlich gilt für die direkte Arbeit mit den XML-Daten: *„Baden strengstens verboten“* in einem Baggersee. Niemand übernimmt Haftung. Doch am Wochenende ist der See voll Betrieb. Die Jugendlichen wissen, so hier alte Bagger unter Wasser eine Gefahr darstellen. Also, man kann und sollte keine Bedienungsanleitungen veröffentlichen „how to deal with ...“ sondern lediglich Umgangshinweise. Otto Normalverbraucher wird sonst die XML-Datei in der Struktur zerstören und sich danach beschweren.

Der Umgang mit solchen XML-Daten ist für den geübten Nutzer aber kein Problem. Auch bei anderen Tools ist es ähnlich. Im Microsoft Visual Studio kann man in den Projektfiles in XML direkt ändern, wesentlich schneller beispielsweise Include-Pfade für alle Compiliervarianten ändern, oder auch die relativen Pfade der Memberfiles. Das geöffnete Visual Studio erkennt sogar ein on-demand geändertes Projektfile und fordert zum Nachladen auf. Dies nur als Beispiel.

Beispielsweise sind relative Pfade zu Bild-Files einfach anpassbar wenn sich das gesamte Bildverzeichnis geändert hat, mittels search & replace. Der Arbeitsaufwand ist wesentlich geringer.

Man sollte immer vorher den bisherigen Stand als Sicherheitskopie speichern und den neuen Stand genau anschauen. Ist doch etwas in der XML-Struktur falsch, dann wird die Stelle angezeigt. Auch hier ist es hilfreich, vor der Änderung in der Content.xml den gegebenen Stand abzuspeichern um vergleichen zu können.

Das Rückspielen einer geänderten content.xml ist einfach möglich durch Speichern im zip-Archiv und dann wieder zurück kopieren als odt (ohne die zusätzliche zip-Extension).

Wer diesen Arbeitsgang nicht verstanden hat, sollte die Finger vom XML lassen.

6 Einfügen von Bildern

6.1 Bilder immer als link

Meine Erfahrung und Herangehensweise seit Jahren ist: Bilder immer ***nur als link einfügen.*** Das ist entgegen oft influenzierten Hinweisen, bitte keinen Link benutzen.

Warum: Die Dokumenten sind nicht fix fertig sondern leben von der Pflege. Auch, teils insbesondere die Bilder werden gepflegt, wenn / weil es sich um Grafiken handelt, die technische Dinge beschreiben. Bildfiles sind häufig auch mehrfach verwendet, beispielsweise in einem HTML, wo sie ebenfalls sowieso grundsätzlich verlinkt sind.

Gibt es nun ein paar neu bearbeitete Bilder, dann genügt im Libre-Office nur das neue öffnen. Oder *Update – All*. Dann sind freilich Widersprüche erkennbar wenn sich die Bildgrößen geändert haben. Das erkennt man aber schnell, kann anpassen, zumindestens ist das Bild in der neuen Erscheinung ohne Arbeitsaufwand erst mal da.

Relative links nutzen:

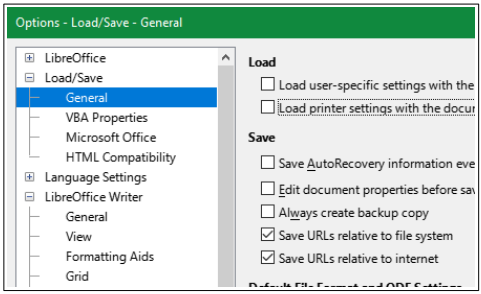

Figure2:LibreO/UseRelativeLinksDialg.png

Absolute Links haben den entscheidenden Nachteil, dass sie nur für die Festplattenorganisation des einen PCs gelten oder es muss eine absolute allgemeingültige Ordnung auf allen Festplatten geben. Für den Ottonormaldokumentenschreiber mit nur einem PC im Gebrauch mögen die absoluten Links passend sein. Für professionelles arbeiten vollkommen ungeeignet. Die Speicherung aller Links im Dokument (intern) als relativ wird eingestellt unter Figure2:

Wie im folgenden Abschnitt 7 Hyperlinks im Dokument als relative Links unter *Nutzung der Symbolic linked directories auf Fileebene, um die links passend zu gestalten* dargestellt, sollten die Bilder zentral bei mir in einem `img`-Directory möglicherweise eben als symbolic linked directory in die Nähe des Dokumentes plaziert werden. Das ist auch wichtig, wenn aus dem odg ein html erzeugt wird und damit die Bilder verlinkt sind. In einem pdf werden sie allerdings immer intern gespeichert.

6.2 Bild-Titel

Die angebotene Möglichkeit „Insert Caption“ im Kontextmenü des Bildes funktioniert schlecht. Probleme sind unklare Rahmen und Positionen, man weiß nicht was los ist.

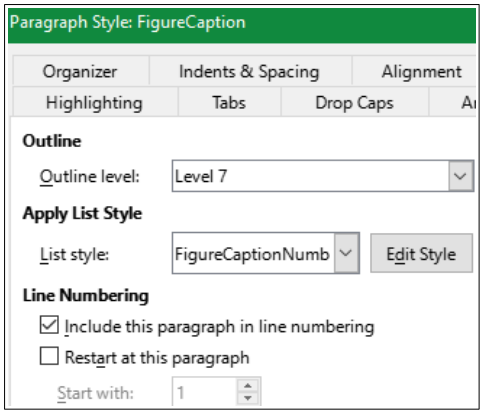

Figure3:LibreO/style_FigureCaption.png

Eine einfache beherrschbare Form besteht darin, ein Absatzformat zu definieren, bei mir als Style *„FigureCaption“*. Dieser

Paragraph-Style ist so ausgerichtet, dass unter *Outline & list* folgendes eingestellt ist, siehe Figure3:

Der List style F*igureCaptionNumb* ist eigens dafür unter *List styles* erstellt (entsprechende Spalte bei Styles in der Sidebar auswählen):

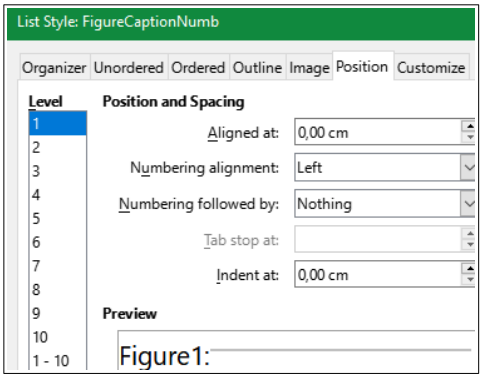

Figure4:styleFigCaptionNumbPos.png

Im Tab Customize wird dann als Vor- und Nachspann *Figure :* eingestellt:

Das Outline level Level 7 in Figure3:hilft, dass die Bilder nicht im Inhaltsverzeichnis mit auftauchen.

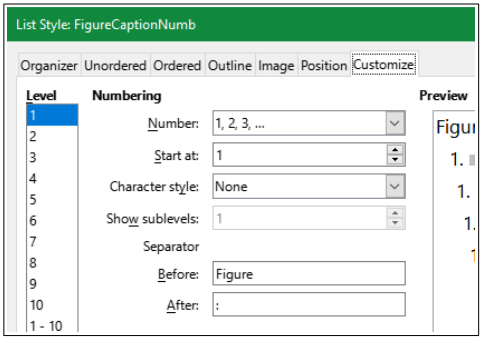

Figure5:styleFigCaptionNumbCust.png

Der Grafik-Style des Bildes ist so ausgelegt, dass das Bild zum Absatz zugeordnet wird, und zwar zu dem Absatz des FigureCaption. Damit kann das Bild mit diesem Absatz, der Bildunterschrift, insgesamt gelöscht, woanders eingefügt usw. werden. Man weiß dass das Bild zum Absatz der Bildunterschrift gehört. Die Position des Bildes ist immer [0,0], es sollte nie verschoben werden.

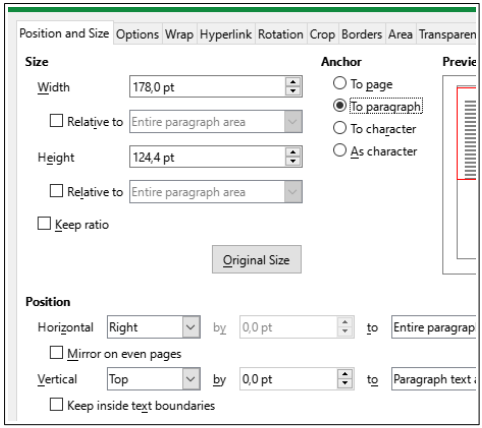

Figure6:LibreO/Option_Image.png

Wenn es Probleme mit der Positionierung gibt, dann muss eben der Bildunterschrifts-Absatz entsprechend woanders positioniert werden. Bild und Unterschrift gehören immer zusammen. Das ist eine klare Regel die bei der Textgestaltung hilft.:

Als textuellen Bild-Titel selbst habe ich immer den Filenamen der Grafik benutzt. Das hilft bei der Pflege des Dokumentes. Für den Leser ist der Filename eventuell auch sprechend, er sollte passend gewählt werden. Wobei, entscheidend ist oft nicht der Text der Caption, sondern der erklärende Text, in dem auf das Bild verwiesen wird. Den Querverweis findet man nach <F2> unter *„numbered Paragraphs“*: siehe Figure3: LibreO/style_FigureCaption.png.

Hinweis: Die filenamen sollten nicht zu lang sein, damit sie in der Caption nicht schon umbrechen. Andererseits sollten sie eben sprechend sein. Das ist eine Herausforderung. Bitte keine Leerzeichen im Filenamen selbst verwenden. In der Caption kann man passend Leerzeichen einführen.

7 Hyperlinks im Dokument als relative Links

Hyperlinks sind sehr oft primär auf eigene Dokumente in der Nähe bezogen. Beispielsweise bei einer Softwarebeschreibung auf automatisch generierte Softwaredokumentation (javadoc) oder parallele Artikel.

Nun ist es zwar fast eine Glaubensfrage, ob man denn auch einen PC ohne Internet betreiben können müsste. Meine Antwort ist klar „ja“, denn es passt alles unmitelbar notwendige auf die lokale Festplatte. Arbeitet man unterwegs und hat mal kein Internet, ist man nicht gleich blockiert.

Damit sind die relativen Links auch hier wichtig. Sie beziehen sich bei Querverweisen also auf Dokumente auf der selben Festplatte in der Umgebung.

Werden dieses Files auf einer Webseite plaziert, dann funktionieren diese ebenfalls als relative Links, innerhab des Host der Webseite.

Nutzung der Symbolic linked directories auf Fileebene, um die links passend zu gestalten:

Relative Links werden wie gegeben in ein pdf übertragen. Wird das pdf woanders gespeichert als das odt, dann gibt es zunächst Probleme. Bei mir ist ein odg unter `Thema/source.wrk/src/Component/odt/file.odt` abgespeichert, gemäß der Regeln beschrieben in vishia.org/SwEng/ html /srcFileTree.html (www).

Das erzeugte pdf ist aber unter `Thema/pdf/subthema/file.pdf` gespeichert.

Um auch das richtige File bei der Bearbeitung des Dokumentes anwählen zu können, wird das `img`-Verzeichnis als ***Symbolischer Link*** unter `Thema/source.wrk/src/img` beispielsweise für Bildfiles zusätzlich plaziert. Ein symbolischer Link ist unter Unix seit je her bekannt (`ln -s`). Unter Windows ist dies mittlerweile (eigentlich seit Windows-Vista) auch möglich, mit

```
mklink /J name D:\path\to\original
```

Das funktioniert manchmal nicht über Netzwerkgrenzen (Windows-Eigenschaft), ist aber gut brauchbar. Der `mklink /D` kann mehr, braucht aber Administratorrechte beim anlegen.

Diese symbolischen Link-Verzeichnisse sind bei den Hyperlinks im Dokument nützlich. Sie müssen einmalig angelegt werden, sie können gelöscht und neu angelegt werden, wenn etwa die Verzeichnisstruktur geändert wird. Man kann also zunächst die Links immer relativ als „Dokumentenlink“ einfügen.

Die Links werden zwar im Dialogfeld absolut angezeigt, doch werden sie relativ verarbeitet.

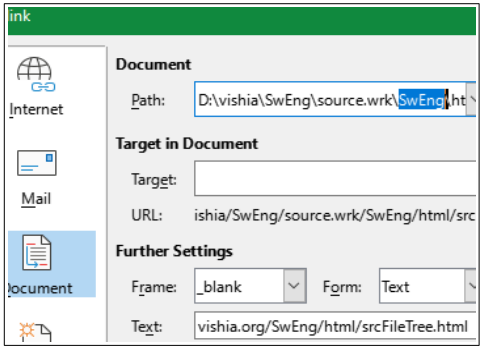

Figure7:LibreO/DialogDoculink.png

Der folgende Link zeigt auf ein html für die Verzeichnisstrukturgestaltung: vishia.org/SwEng/html/srcFileTree.html.

Das in *Figure7:* erkennbare Verzeichnis `D:\vishia\SwEng\source.wrk\`**SwEng** ist ein symbolic linked directory. Der relative Link im XML (siehe chapter *5 Arbeit direkt in der XML-Codierung* lautet:

```
xlink:href="../../../../SwEng/html/srcFileTree.html"
```

Vom aktuellen Dokument wird also 4 Stufen rückwärts gegangen. Eine Stufe davon ist innerhalb des odg auf das odg selbst (quasi aus content.xml). Aus sicht des Dokumentes sind es damit 3 Stufen. Der relative Link wird gebildet im Libreoffice mit Vergleich des eigenen Speicherortes mit dem verlinkten Speicherort. Das Dokument ist hier in

```
D:\vishia\SwEng\source.wrk\src\
LibreOfficeUsage\odt\LibereOfficeUsage_de.odg
```

gespeichert. Die Pfade treffen sich beim gemeinsamen

```
D:\vishia\SwEng\source.wrk.
```

Von dort aus sind es 3 Stufen bis zum odg.

Aus Sicht des erzeugten pdf sind es dann 3 Stufen rückwärts. Das pdf steht auf

```
D:\vishia\SwEng\pdf\LibreOffice\
LibreOfficeUsage_de.pdf
```

Mit den 3 Stufen rückwärts wird hier das Verzeichnis `D:\vishia` getroffen, von dem dann `.../SwEng/html/...` richtig gefunden wird.

Das symbolic linked Verzeichnis `D:\vishia\SwEng\sourc.wrk\SwEng` ist damit nützlich für alle Links, die auf diesen gesamten subdir-tree `SwEng` gehen. Dort sind html-Dokumente, pdfs, images etc. zu diesem Thema zu finden. Soll ein anderes Theme (unter vishia) verlinkt werden, ist ein passendes Verzeichnis wie `D:\vishia\SwEng\sourc.wrk\emC` wieder als symbolic dir notwenig, dass orginal auf `D:\vishia\emC` zeigt, usw.

Das ist eigentlich nicht zu kompliziert und auch gut zu kontrollieren.

Dieser relative Link funktioniert auch auf der Webseite, aber nur dann wenn das pdf im Browser geöffnet wird. Dann gelten die vom Browser auf die Webseite bezogenen daraus ermittelten Pfade.

Wird das pdf dagegen lokal gespeichert, dann hängt es vom Speicherort ab ob die Links gefunden werden. Will man bewusst mit dem downgeloadeten pdf lokal arbeiten, müssen also auch notwendige verlinkte Files geladen werden. Bezüglich einer Javadoc ist diese ggf. sowieso sinnvoll organisierbar, auch bezüglich benachbarter Files. Ist ein File im eigenen Downloadbereich nicht vorhanden, ist der Link eben nicht aufrufbar.

Aus diesem Grund, damit das pdf jedenfalls downgeloaded genutzt werden kann, sind in meinen Dokumenten alle links nochmals als www.links enthalten: Also:

vishia/SwEng/html/srcFileTree.html (www).

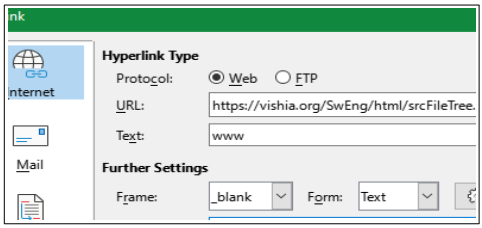

Figure8:LibreO/Dialog_wwwLink.png

Hinter dem Text www verbirgt sich also der direkte Link auf die Webpage. Dieser funktioniert immer wenn ein Internet vorhanden ist.

Bezüglich der Darstellung der Links auch auf Bedienerebene als relative Links gibt es zwei Bug-Einträge:

[1]
https://bugs.documentfoundation.org/show_bug.cgi?id=111957

"Change the way to define relative vs absolute URLs"

[2]
https://bugs.documentfoundation.org/show_bug.cgi?id=128216

"UI: Relative paths should be displayed as relative"

Dass die Links nur absolut angezeigt werden obwohl sie relativ gespeichert sind und relativ funktionieren, ist eine starke Verletzung des Prinzips *What you see is what you **have***.

8 Template

Ein Template mit allen Styles und Beispieltexten ist unter

vishia.org/SwEng/deploy/LibreOfficeTemplates/LibreOfficeTemplate_A5.odt.(www).

gespeichert.

Der folgende Text ist im Template enthalten als Beispiel und als Awwendung aller relavanten Styles, die damit als *„Applied Style“* gefunden werden.

1 Substantiation

This format template contains all styles and some examples for usage.

Copy it under your filename.odg, remove the content and use the styles.

Have a side glance to the examples here.

Saved as

vishia.org/SwEng/deploy/LibreOfficeTemplates/LibreOfficeTemplate_A5.odt.(www).

Note: The style.xml can be dissolved from ....odg.zip and copied in another document, but be carefully with existing styles.

TODO remove stupid styles here!

Note: The style Text-2024-01-30 contains a label, time stamp for the style.

```
  static void testNumeric (TestOrg parent) {
    TestOrg test = new TestOrg("testNumeric formatted", 2, parent);            (1)
    OutTextPreparer otxNumLine = new OutTextPreparer("otxNumLine"              (2)
        , null                                                                 (3)
        , "ix, f1, f2"    //arguments need and used.                          (4)
        , "One line with values at ix=<&ix>: <&f1:%1.3f>, <&f2:%1.3f> \n");   //The
pattern. (5)
    OutTextPreparer.DataTextPreparer otxData = otxNumLine.createArgumentDataObj(); (6)
    StringBuilder sOut = new StringBuilder();
    try {
      for(int ix = 0; ix < 6; ++ix) {
        double f1 = ((float) ix)/2.0;            (7)
        double f2 = Math.sin(f1);
        otxData.setArgument("ix", ix);           (8)
        otxData.setArgument("f1", f1);
        otxData.setArgument(2, f2);
        otxNumLine.exec(sOut, otxData);          (9)
      }
    } catch(IOException exc) {
      test.exception(exc);
    }
    test.expect(sOut, cmpNumeric, 5, "");         (1)
    test.finish();
```

(1) This operation is part of class Test_OutputTextPreparer (=>www)

(2) The working instance of the OutText-Preparer is created here locally. Recommended as final instance variable.

- List with manual bullets
  - List2
    * List3

Text

x List4

- List1_24 with more indent

Illustrative addional explanation, this may be a longer paragraph.

3.1 Sub chapter

1.1.1 SubSub chapter

1.1.1.1 4th Chapter

To show enough content in a diagram you may use an A3 paper in As living example look on the following Class-Object-diagram:

character `ccode-Java` `ccode.Sh` `ccode-VHDL` `ccode-Cpp` `ccode-Marker`

/src/UML_FB_DiagramTemplates/odg/UML_FB_DiagramsTemplate.otg

After beautification it looks like

```
Code
```

```
CODE
```

```
Code-VHDL
```

Figure1: Code-Desciption

```
Code-Sh
```

```
Code-VHDL
```

```
Code-Java
```

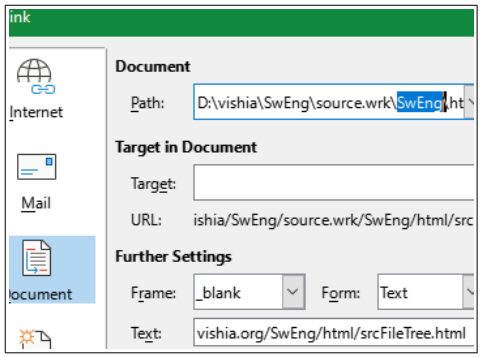

Figure9: DialogDoculink.png

Example for image in columns

Example link relative and www:

vishia/SwEng/html/srcFileTree.html (www).

7 Links and bibliographie

[1] https://en.wikipedia.org/wiki/Unified_Modeling_Language with hint to the founder Grady Booch, Ivar Jacobson and James Rumbaugh

8 Requirements

The Identifier of the **requirements** can be found *in the documentation* and in the tool software. The numbers are unrelated, it's only an id without sorting approaches. The requirements are agil. It means they describe the current state. The Requirements are not sorted here by number, they are sorted by content association. Use ctr-F to find a specific number.

Req740 The outputs of FBlocks can be current values (from the same step time) `ofpVout` or the values from the step time before (state value, adequate unit delay in Simulink) `ofpZout`.