Inhaltsverzeichnis

Kapitel 1

Werkzeugkiste für Self-Publisher und jedermann

Für Autoren, die einmal ein Buch veröffentlichen wollen, ist die Sache relativ einfach: Programme wie Sigil[1] oder LibreOffice[2] können dazu verwendet werden, um die üblichen Zielformate EPUB und PDF zu erzeugen. Doch selbst in diesem einfachen Fall werden schnell die Grenzen deutlich, die sich aus einer allzu manuellen Herangehensweise ergeben, denn nachträgliche Korrekturen können umfangreiche Änderungen erforderlich machen. Erst recht der Umgang mit größeren Textbeständen, sehr spezifische Aufbereitungs-Anforderungen oder typische Aufgabenstellungen bei den Verlegern neuen Typs verlangen Software-Lösungen, die immer wiederkehrende Arbeitsschritte automatisieren und gleichzeitig nicht statisch auf einen einzigen Anwendungsfall zugeschnitten sind, sondern flexibel im Rahmen recht unterschiedlicher Projekte eingesetzt werden können – mit anderen Worten: eine Werkzeugkiste für Self-Publisher, Verlage neuen Typs, alten Typs und jedermann.

Glücklicherweise hat die Technik im Bereich der Medienproduktion in den letzten Jahrzehnten große Fortschritte gemacht, was nicht zuletzt auch Standardisierungsprozessen und der Verfügbarkeit freier Implementierungen zu verdanken ist. Während nun die etablierte Verlagswelt langsam umsteigt auf „Digital First"-Workflows, darf die Self-Publisher-Szene dahinter nicht zurückbleiben, zumal das Veröffentlichungsmonopol ja gerade infolge ebendieser Entwicklungen weggefallen ist und es gilt, das entstandene Vakuum mit einem neuartigen Literaturbetrieb zu besetzen und auszubauen, bevor dieser Leerraum von un- und antidigitalen Angeboten eingenommen wird. Zu beobachten ist dabei, dass all die neuen Teilnehmer im digitalen Literaturbetrieb mehr oder weniger dieselbe Technik brauchen und teils auch schon angefangen haben, jeder für sich Lösungen zu schaffen. Weil man aber niemals um die Technik, sondern um die Bedeutsamkeit von Inhalten konkurrieren sollte, ist es an der Zeit, eine gemeinsame technische Basis zu schaffen, die den Stand des unabhängigen Publizierens insgesamt fördert, anstatt wieder und wieder völlig unnötig in die Lösung bereits gelöster Probleme zu investieren. Natürlich gibt es bereits diverse Projekte, die an einer solchen gemeinsamen Basis oder auch einzelnen

[1] `https://github.com/user-none/Sigil/releases`
[2] `https://de.libreoffice.org/`

Komponenten dafür arbeiten, mit publishing-systems.org[3] sollen diese Bemühungen verzeichnet, bewertet, unterstützt, miteinander verknüpft und ergänzt werden. Es folgt nun eine Beschreibung der Design-Kriterien für unsere eigene Software als auch unseres Projekts an sich.

Der wichtigste Grundsatz lautet **freie Lizenzierung**. Seit der allgemeinen Verfügbarkeit von Digitaltechnologie und der darin inherenten Entkoppelung der Information von ihrem physischen Träger bei gleichzeitig immenser Skalierbarkeit durch den Computer als Universalrechenmaschine (Interoperabilität) steht fest, dass das prä-digitale Urheberrecht in seiner gegenwärtigen Form dringend reformbedürftig ist, weil es mittlerweile infolge technischer, rechtlicher, sozialer und ethischer Probleme das genaue Gegenteil von dem erreicht, wofür es eigentlich gedacht war. Seit den 80er-Jahren tobt der Kampf um digitale Grundrechte in den Bereichen Software, Musik, Video und jetzt auch bei den Büchern, wo aber die Öffentlichkeit wenig organisierte Interessensvertretung hervorgebracht hat und der Gesetzgeber tendenziell eher den Vorschlägen restriktiver Rechteverwerter folgt.

Bei Software und manchen Formaten ist eine Form von DRM schon allein aus technischen Gründen gegeben, sodass die unabhängige, eigenständige Datenverarbeitung effektiv unmöglich gemacht werden kann. Die Anwendung des Urheberrechts auf Software trotz ihres Charakters als Werke von praktischem Nutzen tut ihr Übriges, um Nutzer und andere Entwickler in Abhängigkeit zu bringen. Die einzigste Chance besteht im Moment darin, dass die Urheber ihre Software frei lizenzieren, indem Nutzern umfangreiche Nutzungsrechte eingeräumt werden und anderen Entwicklern die kollaborative Zusammenarbeit ermöglicht wird. Darum ist unsere Software unter der GNU Affero General Public License 3[4] or any later version lizenziert, um die Verfügbarkeit, Veränderbarkeit, Weiterverbreitbarkeit und unabhängige Verwendbarkeit langfristig sicherzustellen.

Wenn es um Medienwerke wie Bücher geht, ist freie Lizenzierung genauso wie bei der zugrundeliegenden verarbeitenden Software eine elementare Voraussetzung für einen ganzheitlich digitalverträglichen Literaturbetrieb. Vielfältig muss das digitale Potential im Hinblick auf den Umgang mit Texten unausgeschöpft werden, weil immer noch das Urheberrecht dazu missbraucht wird, das Geschäftsmodell des Verkaufs einzelner Dateikopien als Nachbildung der Knappheit aus der physischen Welt einzuführen, obwohl der digitalen Welt der Überfluss zugrundeliegt und die künstliche Verknappung dem Anliegen und Zweck einer Veröffentlichung direkt entgegensteht („Veröffentlichung“ hat etwas mit Öffentlichkeit zu tun). Ziel muss es deshalb sein, die Arbeit an und mit frei lizenzierten sowie gemeinfreien Texten aktiv voranzutreiben, die dafür vorhandenen Geschäftsmodelle weiter zu etablieren und so etwas wie eine freie digitale Bibliothek in welcher Form auch immer entstehen zu lassen, die in den Genuss umfangreicher als auch vielfältiger Community- und Softwareunterstützung kommen wird.

Mehr zum Thema von Richard Stallman (über Software[5], über andere Medienwerke[6]), Cory Doctorow (über Implikationen von Digitaltechnologie[7], über

[3] http://www.publishing-systems.org/index.php?lang=de
[4] https://www.gnu.org/licenses/agpl-3.0.html
[5] https://archive.org/details/Richard.Stallman.Manchester.2008
[6] https://www.youtube.com/watch?v=eginMQBWII4
[7] https://www.youtube.com/watch?v=nZFg-uq5zBA

DRM[8]) und The League Of Noble Peers (Steal This Film 2[9]).

Die nächste wesentliche Eigenschaft der Software in der Werkzeugkiste betrifft die **Automatisierung**. Es gibt einen ständig wachsenden Bestand an Texten und jeder dieser Texte bedarf umfangreicher Pflege, Anreicherung und Aufbereitung. Anstatt viel Zeit mit manueller Bearbeitung zu verschwenden, die bei der nächsten Gelegenheit schon wieder hinfällig werden kann, sollen wiederkehrende Arbeitsschritte möglichst automatisiert werden, damit mehr Zeit für automatisierungskompatible Anreicherungen bleibt. Die Automatisierung geht keineswegs zulasten der Qualität oder der Gestaltungsfreiheit, sondern kann im Rahmen der Verarbeitung durchaus Konfigurationseinstellungen oder die Einbindung externer Module zwecks Sonderbehandlung vorsehen, sodass der Workflow ein durchgängig datengetriebener ist und die Automatisierungssoftware lediglich die Methodik zur Verfügung stellt, um von einer definierten Eingabe zur gewünschten Ausgabe zu gelangen, womit der Prozess beliebig oft auf dieselben oder kompatible Eingabedaten neu angewendet werden kann.

Die technische Grundlage dafür stellt XML als universaler Standard zur Auszeichnung von textorientierten Daten dar, womit zahlreiche auf bestimmte Anwendungsbereiche spezialisierte Formate wie z.B. HTML, DocBook, ODT oder TEI definiert wurden, wofür in quasi allen Programmiersprachen Zugriffs- und Verarbeitungsbibliotheken zur Verfügung stehen, worauf mächtige Werkzeuge wie XML-Schema-Validierer, XSLT- und XSL-FO-Prozessoren operieren. XML sichert dabei die Interoperabilität, indem zwischen unterschiedlichen Formaten hin- und herkonvertiert werden kann, Nicht-XML-Quelldateien über Parser nach XML und XML-Daten über entsprechende darauf zugeschnittene Generatoren (evtl. irreversibel) in die gewünschten Zielformate überführt werden. Freilich sind Konzepte wie Single Source Publishing[10] oder gar „Multi Source Publishing“ (Erzeugen mehrerer Zielformate unter der Zusammenfügung mehrerer Datenquellen, die womöglich ihrerseits jeweils mehrfach konvertiert werden müssen) ganz im Sinne der Automatisierung. Allerdings soll die technische Komplexität der Formate, der Verarbeitung und der Abstimmung der Werkzeuge vor dem Benutzer durch grafische Oberflächen komplett verborgen werden (siehe Robert Cailliau zu den Grundlagen des World Wide Webs[11]), weil einerseits manuelle Eingriffe beim nächsten Durchlauf hinfällig wären und andererseits die Erstellung qualitativer Quelldokumente und die Definition von Workflows mithilfe von sinnvollen Standardvorgaben stark vereinfacht werden können.

Die Automatisierung profitiert ganz erheblich von der **Modularität** der Software in der Werkzeugkiste. Die meiste Software für Self-Publisher folgt einer monolithischen Architektur, d.h. ein einziges großes Programm vereint sämtliche angebotene Funktionalität in sich selbst mit dem Ziel, eine integrierte Arbeitsumgebung bereitzustellen. Der Nachteil davon ist, dass einzelne Funktionen oft nicht separat aufgerufen werden können, sondern stattdessen immer die komplette Arbeitsumgebung gestartet werden muss. Dann ist die Arbeitsumgebung in der Regel eine grafische, sodass die Bedienung des Programms primär mit der Maus erfolgt. Bei der Automatisierung macht es aber nur wenig Sinn, Klicks auf Buttons und Menüs auszulösen, denn die Eingabe von Daten

[8] `https://www.youtube.com/watch?v=HUEvRyemKSg`
[9] `https://archive.org/details/StealThisFilmII`
[10] `https://en.wikipedia.org/wiki/Single_source_publishing`
[11] `https://www.youtube.com/watch?v=x2GylLq59rI`

soll ja schließlich von außen parametrisiert angesteuert werden, um einen rein datengetriebenen Ablauf zu erreichen.

Dementsprechend soll die Software, die im Rahmen des Projekts entwickelt wird, aus einer Reihe von kleinen Einzelprogrammen bestehen, die zu größeren automatisierten Workflows zusammengeschaltet werden können. Einerseits entsteht dadurch eine hohe Flexibilität, weil die Programme auch in anderen Kombinationen eingesetzt werden können, andererseits können zwischen den einzelnen Verarbeitungsschritten externe Komponenten aufgerufen werden. Dass die Programme auch abseits automatischer Workflows zur Bewältigung begrenzter Aufgaben in einem ansonsten manuellen Aufbereitungsworkflow genutzt werden können, versteht sich von selbst. Dieser Ansatz schließt übrigens keineswegs aus, dass vor oder während der Verarbeitung umfangreiche benutzerspezifische Einstellungen hinterlegt werden können, die dann von den jeweiligen Einzelschritten berücksichtigt werden. Eine optionale grafische Oberfläche ist dabei behilflich, die überdies auch zur Steuerung und Bedienung der Einzelprogramme, der Workflows und des Gesamtsystems dient. Um die Einzelprogramme möglichst unabhängig von den sie aufrufenden Workflows zu halten, erfolgt die Kommunikation über wohldefinierte Schnittstellen, über welche die Workflows die in „Jobs“ zusammengefassten Einstellungen einspeisen können. Ein Nachteil dieses Ansatzes ist, dass die Komplexität der Abhängigkeiten bei Änderungen an den Schnittstellen oder in den Einzelprogrammen umso mehr steigt, je mehr Workflows sich der betroffenen Programme bedienen – dem soll nach Möglichkeit mit zusätzlichen Abstraktionsebenen begegnet werden, welche Schnittstellendetails nach oben hin verbergen (kapseln).

Fortsetzung folgt.