

Quo vadis? Der beschwerliche Weg mit XML-Technologien

Arnold, Eckhart

arnold@badw.de

Bayerische Akademie der Wissenschaften, Deutschland

ORCID: 0000-0002-1138-499X

Müller, Stefan

mueller@badw.de

Bayerische Akademie der Wissenschaften, Deutschland

Raaf, Manuel

raaf@badw.de

Bayerische Akademie der Wissenschaften, Deutschland

Einleitung

Die XML-Idolatrie, wie sie im ersten Jahrzehnt unseres Jahrhunderts herrschte, ist besonders außerhalb der Geisteswissenschaften einer gesunden Ernüchterung gewichen, wo XML vielfach durch andere Serialisierungsformen ersetzt worden ist (JSON, HTML5, Sphinx (anstelle von DocBook), CSS (statt XML-FO)). Innerhalb der Geisteswissenschaften freilich ist die Verwendung von XML ungleich lebhafter geblieben, was seinen guten Grund darin hat, dass für Geisteswissenschaften wohl mehr als für irgendeinen anderen Bereich der semantisch gegliederte Fließtext überragende Bedeutung hat und dass sich XML für nichts so sehr eignet wie für die Serialisierung von eben diesem. Doch liegen schon in diesen Verhältnissen auch die Urgünde dafür, dass der Weg mit XML-Technologien ein manchmal allzu beschwerlicher werden kann:

Was nur oder vornehmlich für Geisteswissenschaften von Interesse ist, bildet eine recht kleine und wirtschaftsschwache Nische, die für Entwickler:innen außerhalb der Geisteswissenschaften von vergleichsweise geringem Interesse ist; und innerhalb der Geisteswissenschaften kommen Entwickler:innen vergleichsweise seltener vor als in der Mathematik und den Naturwissenschaften. So lässt die Werkzeuglandschaft aus geisteswissenschaftlicher Sicht jedenfalls zu wünschen übrig: Wer semantisch strukturierten Text eingeben will, findet dafür keinen so gebrauchsfertigen Editor wie es eine der üblichen WYSIWYG-Anwendungen wäre (die aber nicht semantisch strukturiert serialisieren).

Da XML für Geisteswissenschaften so nützlich und bedeutend ist, hat sich dort wohl am hartnäckigsten eine Neigung erhalten, XML und XML-Technologien noch über deren Eignung hinaus zu verwenden. Das ist umso beden-

licher eingedenk des ersten Punktes; denn so treffen übermäßige Anforderungen auf eine unterversorgte Werkzeuglandschaft.

Im Folgenden werden die sich daraus weiter ergebenden Schwierigkeiten eingehender sowie anhand von Beispielen aus der Praxis beleuchtet und ebenfalls praktisch erprobte Abhilfen zur Milderung mancher Schwierigkeiten angedeutet.

XML, XQuery, XSLT und XML-Datenbank

Über den ursprünglichen Einsatzzweck hinaus wird XML zum Beispiel als ein Ersatz für Datenbankformate betrachtet, wenn es um die Erarbeitung eines Wörterbuches geht (vgl. Müller-Spitzer, 2007). Begründet ist dies durch den vermeintlich leichten Einstieg, da XML ein menschenlesbares Format ist, das anhand weniger Regeln frei gemäß den individuellen Bedürfnissen definiert werden kann. Doch ist XML für diesen Einsatzzweck nicht gedacht (vgl. Harold, 2003, 230f.). Die vermeintliche Einfachheit erzeugt zudem zwangsläufig Probleme, denn Datenintegrität und -redundanz sowie (mathematische) Logik sind wichtige und keineswegs einfache Themen, die bei der Datenmodellierung und auch –eingabe meist wenig beachtet werden. XML wird schnell unhandlich und die verfügbaren Werkzeuge sind meist zu speziell oder zu dürftig. Die Folge: Ohne Technikerunterstützung ist eine semantische Text- oder Datenauszeichnung nicht möglich und wird daher auch künftig in vielen geisteswissenschaftlichen Projekten unterbleiben.

Die Transformationssprache XSLT wird in XML-Syntax geschrieben und regt damit all jene an, die bereits XML, aber noch keine Programmiersprache kennen, Skripte zur Umwandlung von XML nach z. B. HTML zu schreiben. Doch auch diese Sprache sowie die wenigen XSLT-Prozessoren, mit denen XSLT-Skripte ausgeführt werden, weisen zum Teil immense Schwächen auf, die insbesondere in Kombination mit den oben erwähnten Problemen auf Datenebene die Weiterverarbeitung oft unnötig kompliziert gestalten.

Erschwerend hierbei ist, dass es sich beim XML-Ökosystem um eine Nischentechnologie handelt, die in den frühen 2000er Jahren ihre Blütezeit hatte und seither im Wesentlichen nicht mehr weiterentwickelt wurde, was Unzulänglichkeiten von Werkzeugen und Datenbankumgebungen zur Folge hatte. Beispiele dafür sind die seit mehr als 20 Jahren bestehenden Probleme der Sprachen XQuery und XSLT (vgl. Raaf, im Druck), die dediziert für die Abfrage und Weiterverarbeitung von XML-Dokumenten entwickelt wurden und hierfür oft nicht ausreichen.

Unsere Erfahrungen im „Bayerischen Wörterbuch“ (BWB) sowie in „Bayerns Dialekte Online“ (BDO) zeigen, welche Herausforderungen beim Einsatz von XML-Technologie entstanden und wie sie gelöst wurden bzw. bisher nicht zu lösen waren. Wir werden sie im Vortrag mit entsprechenden Beispiel veranschaulichen. Für das BWB

lag der Fokus hierbei auf der Umstellung des Workflows von Word hin zur Erfassung der Wörterbuchartikel direkt in XML mithilfe des proprietären Editors Oxygen (vgl. Raaf/Welsch, im Druck). Diese verlief nicht ohne Hürden und Unsicherheiten bezüglich der Druckausgabe. Letztlich gelang der Umstieg bei der Dateneingabe allen Mitarbeiterinnen und Mitarbeitern, die Weiterverarbeitung hingegen gestaltete sich in mehrerlei Hinsicht als zeitaufwändige Herausforderung: So gibt es viele Sonderregeln für den Druckexport, die durch eine klarere Auszeichnung im XML hätten vermieden werden können. Darüber hinaus können aus ähnlichen Gründen in der Online-Ausgabe manche Inhalte nicht wie gewünscht dargestellt werden, da sie in Attributen notiert sind und diese in XML naturgemäß keine Ordnung besitzen. In den gegebenen Fällen kann weder der Inhalt noch die unmittelbare Umgebung (d.h. Eltern-Knoten o.ä.) helfen, zu unterscheiden. Es besteht daher keine technische Möglichkeit innerhalb des XML-Ökosystems, die gewünschte Reihenfolge für die Ausgabe einzuhalten. Des Weiteren stellen zunächst trivial wirkende Leerzeichen, Zeilenumbrüche und Tabulatoren ein Problem dar, da sie durch automatische Umbrüche und Einrückungen in Oxygen und/oder durch manuelle Fehleingaben falsch platziert wurden. Zwar herrscht in der XML-Welt die Ansicht, dass Leerzeichen keine Semantik tragen. Faktisch jedoch werden sie lediglich durch Funktionen innerhalb von XPath, XQuery und XSLT normalisiert oder entfernt. Im Dokument selbst tragen sie die Bedeutung des jeweiligen Zeichens. Stehen sie aus menschlicher Sicht an falscher Stelle, beeinträchtigen sie auch nach etwaiger Normalisierung das erwünschte Ergebnis der Ausgabe. Für die Weiterverarbeitung stellt dies einen nicht zu unterschätzenden zeitlichen Mehraufwand dar.

Nichtsdestotrotz ist XML ein Format, das aus unserer Domäne nicht mehr wegzudenken ist und das bei aller berechtigter Kritik auch eine gute Wahl für manche Anwendungsfälle darstellt – für manche, nicht für grundsätzlich alle. Die Einarbeitung in XML-Technologie erfordert zwar deutlich weniger Zeit, als dies bei Alternativen (z.B. SQL, diverse Binärformate, DSL) der Fall ist, doch stellt sich aus den genannten Gründen die Frage, ob XML, geschweige TEI-XML, weiterhin ein generell geeignetes, in nahezu jedem Projektantrag gefordertes Format für alle Belange der digitalen Geisteswissenschaften sein kann. Falls die Wahl auf XML fällt, so muss bereits während der Datenmodellierung sowie späteren Modifizierungen derselben die gewünschte Ausgabe bedacht werden, da andernfalls Probleme auftreten können, deren Behebung viele Ressourcen einfordern. Für die Weiterverarbeitung sollten zudem Alternativen bedacht werden, wie der Wechsel auf eine eigene Lösung in einer Skriptsprache oder der Austausch des primär eingesetzten und in Java geschriebenen XSLT-Prozessors *Saxon* mit seiner nach C++ portierten Variante *SaxonC* (vgl. Raaf/Welsch, im Druck).

Eine reichere Werkzeuglandschaft mitnutzen: HTML5-XML, CSS, JS und Visual Studio Code (“Εὕρηκα”)

Eine mögliche Milderung der eingangs genannten Grundschwierigkeiten ist, eine reiche und nicht auf die Nische der digitalen Geisteswissenschaften beschränkte Werkzeuglandschaft mitzunutzen. Nahe liegt die Landschaft der Netzseitenentwicklung: Erstens kann der Netzseitenstandard HTML5 als XML verwendet werden, sieht man von zwei nur oberflächlichen Einschränkungen ab; man braucht also nicht auf XML zu verzichten. Zweitens ist diese Landschaft besonders reich an weit entwickelten und gut gepflegten Werkzeugen. Drittens und viertens gilt, wenn das Einzugebende endlich ohnehin im Netz zu veröffentlichen ist: Eingebende sehen und kontrollieren dabei viel unmittelbarer das Endergebnis; und man kann einiges an Entwicklungsaufwand einsparen: nicht nur, weil man mehr fertige Werkzeuge vorfindet, sondern auch, weil Transformationen entfallen oder einfacher sind und weil alles unnötig wird, was man nur für eine Anzeige in der Entwicklungsumgebung bräuchte, während fast alles, was man in Richtung auf die Netzseite hin entwickelt, doppelt genutzt wird.

Eingesetzt wird dieser Ansatz (“Heureka” genannt) bei den neuen Vorhaben “Otloh” und “Schelling. Nachlass-Edition” (vgl. Müller 2022). Er wurde für Letzteres entwickelt und besteht darin, mit dem Editor Visual Studio Code samt zweier Extensionen (Syntaxprüfung und Vorschau) sowohl netzseitenkompatibles als auch semantisches HTML5-XML einzugeben und im Hintergrund CSS und JS bereitzustellen, die sowohl für die Vorschau im Editor als auch für die Anzeige im Browser genutzt werden. So konnten mit einer ganz enormen und mangels Kapazitäten auch nötigen Aufwands- und Zeitersparnis alle Anforderungen erfüllt werden: die Eingabe unmittelbar in einer semantisch strukturierten Form, die sich trivial in TEI-XML umwandeln lässt, unterstützt von Vervollständigungen, Syntaxhervorhebung und -prüfung; eine sich stets aktualisierende, Seite-an-Seite mitlaufende Vorschau auf den gerenderten Text; Einbettung komplexerer mathematischer Terme in TeX, die in Vorschau und Browser mitgerendert werden; Druck von Korrekturfahnen; Betriebssystemunabhängigkeit; Quelloffenheit; möglichst ohne den Aufwand, etwas erwerben und abrechnen zu müssen. Zudem: Wegen einer großen Nutzermenge sind alle Lösungsteile ausgiebig getestet und versprechen, eine genügend lange Zeit gut gepflegt und weiterentwickelt zu werden.

Statt XML: LaTeX

Ein weiterer Ansatz zur Minimierung des technischen Betreuungsaufwandes besteht darin, ein herkömmliches, bewährtes Werkzeug für die Edition, z.B. Classical Text Editor (CTE) oder LaTeX zu verwenden.

Die Nutzung beider Werkzeuge können sich auch Nicht-Techniker gut im Selbststudium aneignen. Insbesondere für LaTeX gibt es darüber hinaus eine reichhaltige Einführungsliteratur, wenn sie auch überwiegend auf (moderne) wissenschaftliche Arbeiten und nicht auf die Erstellung klassischer Editionen zugeschnitten ist. (Für letztere vgl. Dunning, 2020.) Von der ebenfalls großen Anzahl guter Eingabewerkzeuge mit Vorschaufunktion profitieren letztere aber ebenso.

Aus LaTeX lässt sich dabei ein semantisch ausgezeichneter Text eher gewinnen als mit CTE. LaTeX erzwingt zwar nicht, ermutigt aber zur semantischen Auszeichnung, so dass Leute, die mit LaTeX arbeiten, das Prinzip *avant la lettre* schon kennen und verstehen. Beiden Werkzeugen ist gemein, dass man den Drucksatz schon mitgeliefert bekommt. Die Konvertierung von LaTeX nach XML ist allerdings nicht trivial. Es gibt eine Reihe verfügbarer Lösungen, aber man kann davon ausgehen, dass keine davon ohne projektspezifische Anpassungen einsatzbereit ist. (Hier wirken sich die vielfältigen Variationsmöglichkeiten von LaTeX ähnlich aus wie die von TEI-XML!)

Der XML-Export von CTE taugt nicht als semantisch ausgezeichnetes Dokument. Hier ist jedoch eine andere Strategie denkbar: Extrahiere den reinen Editionstext ohne Anmerkungen und Apparate, denn – hier die Argumentation – Anmerkungen und Apparate sind im Wesentlichen nur für die menschlichen Leserinnen und Leser von Interesse, müssen also nicht semantisch ausgewiesen und ausgezeichnet werden. Für die maschinelle Rezeption ist umgekehrt der reine Editionstext (mit bestenfalls schlanker Auszeichnung) in den meisten Fällen sogar geeigneter als die volle semantisch ausgezeichnete Edition!

Maßgeschneiderte Auszeichnung und Werkzeugkette beim MLW

Schließlich kann man auch eine „domänenspezifischen“ Eigennotation (DSL) entwickeln, die eine sehr bequeme Eingabe mit im Prinzip jedem beliebigen Editor, der farbliche syntaktische Hervorhebung unterstützt, ermöglicht. DSLs erfreuen sich in der IT-Industrie seit ca. 15 Jahren einer ungebrochenen Beliebtheit (Vgl. Fowler/Parsons 2010) und haben insbesondere im Bereich der Software-Dokumentation mit Sprachen wie Markdown (insb. MyST), reStructured Text und ASCII-Doc, XML-basierte Lösungen wie DocBook und sogar LaTeX weitgehend abgelöst.

An der BAdW haben wir diesen Ansatz beim Mittellateinischen Wörterbuch (MLW) gewählt. (Vgl. Arnold 2019) Nach unseren Erfahrungen ist der Entwicklungs- und Einführungsaufwand bei einer DSL relativ groß und lohnt sich eher für Langzeitprojekte. Insbesondere die Ausarbeitung der formalen Grammatik für eine DSL ist kompliziert und erfordert einige Erfahrung. Dafür erlaubt eine DSL aber ein relativ bequeme Dateneingabe bzw. Text-Codierung.

Da das daraus abgeleitete XML nur noch maschinell verarbeitet wird, muss bei dessen Ausgestaltung auch nicht

Rücksicht auf die Bedürfnisse menschlicher Bearbeiter genommen werden. So kann z.B. auf gemischte Tags, die Text und weitere Tags enthalten können, ganz verzichtet werden. Das erleichtert die Weiterverarbeitung. Dadurch fällt insbesondere der umfangreiche Test-Code für das MLW sehr viel übersichtlicher aus als mit XML.

Die beim XML-Ansatz für das BWB beschriebenen Probleme bei der Weiterverarbeitung tauchten nicht auf.

Bibliographie

Arnold, Eckhart. (2019) Beschreibung der MLW-Notation, auf: t1p.de/uk4vy

Dunning, Andrew. "Review of 'Reledmac. Typesetting technology-independent critical editions with LaTeX'." *RIDE* 11 (2020). doi: 10.18716/ride.a.11.1.

Fowler, Martin und Rebecca Parsons. (2010) *Domain Specific Languages: A detailed guide on implementing both internal and external DSLs*. Boston Munich: Addison-Wesley.

>Harold, E.R. (2003) *Effective XML: 50 specific ways to improve your XML*. 1. printing. Boston Munich: Addison-Wesley (Effective software development series).

Müller, Stefan. (2022) *Heureka-Regelwerk*, auf: schelling.badw.de/auszeichnungsregeln

Raaf, Manuel und Ursula Welsch. (im Druck): *Wannst einischaugst, findest ois ... Die digitale Einigung von Bayern, Franken und Schwaben*. In: Almut König/Mechthild Habermann (Hgg.): *Großlandschaftliche Dialektwörterbücher zwischen Tradition und Innovation*. In: *Namenphilologie trifft Dialektlexikographie. Beiträge aus dem Arbeitskreis für Bayerisch-Österreichische Namenforschung und dem Netzwerk der großlandschaftlichen Dialektwörterbücher - LexikoNet* (=Jahrbuch der Johann Andreas Schmeller Gesellschaft 2022). edition vulpes: Regensburg.

Raaf, Manuel. (im Druck): *Murphy's Law in Electronic Lexicography: what can go wrong, will go wrong. Frequent challenges on the example of Bavaria's Dialects Online*. In: Stöckle, Philipp / Wahl, Sabine (eds.): *Lexicography and Language Variation*. Wien: V&R unipress (Wiener Arbeiten zur Linguistik).

Müller-Spitzer, C. (2007) *Der lexikografische Prozess: Konzeption für die Modellierung der Datenbasis*. Tübingen: Narr (Studien zur deutschen Sprache, Bd. 42)