Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 1935

D2d — Kreatives Schreiben von XML-codierten Texten

Markus Lepper[1] Baltasar Trancón y Widemann[2]

1 Anwendungskontext und Motivation

Der XML Standard [Br06] hat sich mittlerweile zur Codierung von formalen oder semiformalen Texten weithin durchgesetzt. Die Vorteile dieser Standardisierung sind die Wiederverwendbarkeit einer unübersehbaren Fülle von Transformationswerkzeugen und die Unbhängigkeit von proprietären Standards bei Erstellung, Speicherung und Zugriff.

„Mark-Up“ in Texte einzuführen kann in sehr unterschiedlichen Kontexten sinnvoll sein, die weit über das bis dato übliche hinausgehen. Dies aber auf eine jeweils sehr unterschiedliche Art und Weise. So kann es sinnvoll sein, Referenzen auf Personen in mehrbändigen Fantasy-Novellen gleich beim Erstellen, oder in einem ersten redaktionellen Durchgang zu markieren. Ähnlich kann die Markierung von Ortsangaben in Wanderführern, von chemischen Formeln in pharmakologischen Texten, Tempoangaben in Sportreportagen, Tageszeiten in Laborberichten, Zutatenlisten in Kochbüchern etc. den Wert des Elaborates allein durch die mögliche automatische Nach- und Weiterverarbeitung deutlich erhöhen, bis hin zur Präsentation als „dynamisches Dokument“.

Allerdings ist das übliche Serialisierungs-Format von XML keinesfalls geeignet, hingeschrieben zu werden von Autoren im kreativen Fluss des Schreibens. Selbst, ja gerade „syntaxgesteuerte Editoren“ sind nicht in der Lage, Menschen, für die der Akt des Schreibens ein direkter, kontinuierlicher, ja, intimer ist, für Mark-Up zu begeistern.

2 Designprinzipien von D2d

Diese Lücke will D2d schließen. Die Abkürzung steht für „Direct Document Denotation“ oder auch „Directly To Document“ und wird meist „triple-dee“ gesprochen. D2d ermöglicht es einem Autor, mit nur einem ausgezeichneten Befehls-Zeichen im gewohnten Verlauf des Schreibens lesbare aber korrekte, streng typisierte XML-konforme Dokumente zu erzeugen. Insbesondere abstrahiert D2d von dem redundanten Gebrauch von Sonderzeichen und der idiosynkratischen Unterscheidung von „Element“ und „Attribute“ und erlaubt den Permutationsoperator mit beiden.

Der gewohnte Texteditor, die gewohnte Arbeitsumgebung, die gewohnte Methodik kann beibehalten werden. D2d kann mit Kreide und Wandtafel, Papier und Bleistift einfach

[1] semantics GmbH, Berlin, post@markuslepper.eu
[2] Technische Universität Ilmenau, baltasar.trancon@tu-ilmenau.de

notiert werden und ist geeignet für Sprachein- und Ausgabe, und schon durch die einfache Zeichenstruktur behindertenfreundlich.

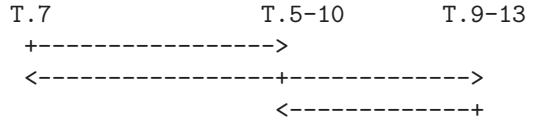
Dass je Arbeitsfeld unterschiedliche Definitionen von Text-Elementen, deren Tags und deren Inhalte getroffen werden müssen, und diese Elemente nicht unmittelbar eine optische Erscheinung ihres Inhaltes bestimmen, sondern vielmer dessen inhaltliche Rolle im Körper des Gesamttextes, hat dazu geführt, dies „semantisches Mark-Up“ zu nennen, im Gegensatz zum „physikalischen“. Das kann in die Irre führen, denn allemal ist es lediglich *Syntax*, was von D2d erkannt und verwaltet wird. Richtig jedoch ist die wichtige Unterscheidung zwischen dem erstellten Text als seiner „Substanz“ und allen möglichen Verarbeitungen und Darstellungen als seiner „Erscheinung“, – ein im Bereich des Digitalen noch weithin unerforschte Dichotomie.

Die Definition von D2d und die vorliegende Implementierung beinhalten nur einfache bekannte Techniken und kleine, unspektakuläre Erfindungen, – das Zusammenwirken allerdings stellte sich als sehr komplex heraus, so dass fast zehn Jahre Entwicklung und Erprobung in das System geflossen sind. [LTW01, LT11]

Grundlegende Design-Eigenschaften sind, (a) die Verwendung offener Standards, wann immer vorhanden. Ausserdem (b) eine grundlegende Orientierung am Anwender, der oft Fach-Experte und kein Informations-Experte ist. Dies z.B. durch „Erste Klasse“-Unterstützung von polyglottem Dokumentationstext, durch graphische „Syntax-Diagramme“ im WIRTH-Stil, durch extensive Fehlerdiagnose und durch einen (ein- und ausschaltbaren) robusten Übersetzungsmodus, der Vorschläge für fehlende Strukturen ergänzt und überflüssige zu deutlich sichtbarem Kommentar neutralisiert.

3 Beispiel

Ein repräsentatives Beispiel ist eine große musiktheoretische Analysearbeit, die als ein *dynamisches* Dokument vorliegt, und nur als ein solches verlustlos vorliegen kann [Le15]. Darin werden u.a. Takt- und Tonart-Angaben mit Mark-Up versehen. In der Nachbereitung („Rendering“) des eingegebenen Textes werden z.B. Navigationselemente erzeugt, die auf das erste, letzte, vorige und folgende Auftreten derselben Tonart verweisen. Im Falle von Taktangaben werden hingegen auf diese Weise alle Bereiche verbunden, die eine nichtleere Schnittmenge haben:



Diese Art der Navigation ist die *dieser konkreten Lese-Situation adäquate*, und mit vorgefertigen, festgelegten Standard-Mitteln nicht zu erreichen.

Ein ganz anderes Beispiel ist die Dokumentation von D2d selber, als Teil der Sammlung `metatools`, siehe [met]. Die Quelltexte aller dieser Seiten sind erreichbar durch das al-

lerletzte auf ihnen enthaltene Link. Man beachte besonders die Notenbeispiele, deren verschieden kombinierte Schichten auch durch XML-Mark-Up unterschieden und dann durch XSLT zwecks Rendering rekombiniert werden.

Wir vermuten, dass nur durch einen derart spezialisierten, dezidierten und inhaltlichen Gebrauch des Rechners Fortschritte auf dem Gebiet der „Digital Humanities“ erst wirklich möglich werden.

4 Parsieren von Tags und Character Data

D2d unterscheidet zwei Ebenen von Parsierung:

Auf der groberen Ebene der „Tag-Parsierung“ werden die eröffnenden Tags, eingeleitet durch das eine, erwähnte Kommandozeichen, in den Text eingestreut. Schließende Tags werden durch einfachste LL(1)-Parsierung inferiert, können aber auch explizit gesetzt werden.

Auf der feineren Ebene der „Character-Parsierung“ werden Parser definiert, die auf einzelnen Zeichen operieren. Die dabei gemeinte Tags werden von dem erkannten Parser automatisch in das Parsierungsergebnis eingefügt. Diese Parser-Ebene wird verwendet, um kleine, domänen-spezifische Entitäten ohne syntaktischen Ballast hinschreiben zu können, wie o.e. Takt- und Tonartangaben, oder Aktenzeichen, Kfz-Kennzeichen, Kalenderdaten, Personen-Namen, chemische Formeln, etc.

Die Definitionen und Transformationen sehen dann aus wie ...

```
chars tk =  [von (S:digit)~+] ~ ([bis "-" (S:digit)~+] )?

siehe auch vorher schon #tk 3-5  oder #tk 7

siehe auch vorher schon
  <tk><von>3</von><bis>5</bis></tk> oder <tk><von>7</von></tk>
```

Die beiden Ebenen von Parsierung funktionieren dezidiert unterschiedlich: die Zeichen-Parser werden nicht-deterministisch ausgewertet und arbeiten „gierig“. Ihre automatische Übersetzung in DTD-Definitionen ist deshalb nicht immer möglich oder zweckmäßig. Ausserdem eignen sich sich nur für relativ kleine Eingaben. Die in der Standard-Distribution vorgesehene MathML-Unterstützung ist die z.Zt. größte bekannte Anwendung und gerät schnell an die Grenzen der Praktikabilität. Da müssen evlt. im Verlaufe der weiteren Entwicklung intelligentere Parsierungstechniken eingesetzt werden.

5 Moduln, Parametrisierung und Rewriting

Die Notwendigkeit einer je Domäne speziellen Lösung setzt meistens voraus (a) das Vorhandensein einer Grundarchitektur, da ja nicht alle Komponenten eines Textes *ex ovo* neu

definiert werden sollen, und in diesem Falle weiterhin (b) einen Mechanismus zur Parametrisierung zwecks Anpassung dieser Grundarchitektur.

D2d verwendet ein Modul-System. Ein Modul dient zunächst zur Abgrenzung von Gültigkeitsbereichen von Definitionen. Erwähnte Grundarchitektur heißt `d2d_gp`. Sie ist gegeben als Sammlung von Moduln und beschreibt eine Textstruktur, die sich an „LaTeX article“, HTML u.ä. orientiert.

Man beachte, dass die gesamten Definitionen ca. 50 Druckseiten beinhalten, darin enthalten der XSLT-Quelltext für die vollständige Übersetzung in das XHTML-Backend, und die (fast !-) vollständige Benuterzdokumentation in Englischer Sprache. (Man vergleiche das mit der Spezifikation „OOXML“ von Microsoft mit ihren angeblich sechstausend Seiten!-)

Zwecks Anpassung kann ein Modul mehrfach importiert werden, wobei es unterschiedlichen *Termersetzungen* unterworfen wird. Die Menge der bei einer zukünftigen Wiederverwendung in einem noch unbekannte Kontext auszutauschenden Definitionen ist (natürlicherweise) nicht bekannt. Deshalb ist in D2d potentiell jede Referenz ein Modulparameter, und ein freies Rewriting kann auf jedes zu importierende Modul angewandt werden und beliebige Referenzen ersetzen, – im Gegensatz zu dynamischer Bindung aber mit kalkulierbaren Auswirkungen.

Die Konsequenzen dieses einfachen Ansatzes sind allerdings recht komplex, und unsere Forschung ist noch im Gange (z.B. bzgl. Terminierung!-). Besondere Beachtung verdient dabei der Fall, dass eine Reifikation von transformierten Definitionen notwendig wird. Dies ist z.B. der Fall wenn Benutzerdokumentation generiert werden soll; dann müssen unterschiedliche Parametrisierungen mit ununterscheidbarem Effekt möglichst unifiziert werden, um die Information an den Benutzer so schlank wie möglich zu halten.

6 Dokumentation der Definitionsmoduln

Die Generierung von Dokumentation reist Erster Klasse: mit jeder Definition kann ein Dokumentationstext in verschiedenen Sprachen angegeben werden. Dieser benutzt eine Variante des Standardformates `d2d_gp`, mit allen Möglichkeiten von Verlinkung und Präsentation.

Aus diesen Texten baut die Implementierung einen XHTML-Text zusammen, der ergänzt wird durch den automatisch generierten Benutzungs-Graphen, der wiederum als Liste von Sprungzielen und als interaktive SVG-Graphik präsentiert wird. Aus den textuellen Termen der regulären Ausdrücke werden WIRTH-Diagramme erzeugt, die wiederum „anklickbar“ sind, etc.

Als Beispiel diene die Dokumentation von `d2d_gp` selber, zu finden am Ende der Tool-Dokumentation [met].

7 Unterstützung von XSLT

D2d unterstützt XSLT in zweierlei Hinsicht: einmal gibt es für jedes definierte Eingabeformat „M“ einen „XSLT-Modus“ des Parsierens, in dem nicht Texte des Typs M parsiert werden, sondern XSLT-Programme, die Texte des Typs M hervorbringen. Der Inferenzmechanismus, der es erlaubt, mit minimalem Mark-Up M-Dokumente hinzuschreiben, wird modifiziert verwendet um zwischen den sog. „Target-Elementen“ und den XSLT-Befehlen zu unterscheiden, – es arbeiten quasi zwei Parser als Ko-Routinen. Dies erlaubt XSLT hinzuschreiben, das schon fast wie eine „richtige Programmiersprache“ aussieht:

```
#template #match a:tableOfContents
  #call makeVisibilitySwitch
    #arg id-switch #xp'_switch_TOC'
    #arg id-table  #xp'__div_TOC'
  #p#id __div_TOC
    #a #id  TOC  #span #class tocTitle #apply #xp a:caption#/a
    #br/
    #choose
      #when /descendant::a:part
        #apply #xp /descendant::a:part  #mode maketoc
      #other
        #apply #xp /descendant::a:h1 #mode maketoc
  #/p
```

Nebenbei erreicht diese Eingabemethode mit einfachsten Mitteln auch einen großen Schritt in Richtung typkorrekter XSLT Transformationen, wie beschrieben in [LT15].

Zum zweiten kann gleich bei der Definition eines D2d-Moduls eine Menge von Übersetzungsregeln mitgegeben werden, und zwar für alle Kombinationen von neu definierten Element-Arten und verschiedenen Ausgabeformaten („Backends“). Beim Prozessieren eines Eingabetextes werden dann für jedes ausgewählten Backend diese XSLT Regeln kombiniert und angewandt.

Literatur

[Br06] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C.M.; Maler, Eve; Yergeau, Fran cois; Cowan, John: . Extensible Markup Language (XML) 1.1 (Second Edition). W3C, `http://www.w3.org/TR/2006/REC-xml11-20060816/`, 2006.

[Le15] Lepper, Markus: Gustav Mahler, Dritte Sinfonie, Erste Abtheilung. Senzatempo, 2015. `http://senzatempo.de/mahler/gmahler_sinf3_satz1.html`.

[LT11] Lepper, Markus; Trancón y Widemann, Baltasar: d2d — a Robust Front-End for Prototyping, Authoring and Maintaining XML Encoded Documents by Domain Experts. In (Filipe, Joaquim; J.G.Dietz, Hrsg.): Proceedings of the International Conference on Knowledge Engineering and Ontology Delelopgment, KEOD 2011. SciTePress, Lisboa, S. 449–456, 2011.

[LT15] Lepper, Markus; Trancón y Widemann, Baltasar: A Simple and Efficient Step Towards Type-Correct XSLT Transformations. In: Proceedings 26th International Conference on Rewriting Techniques and Applications (RTA 2015). Jgg. 36 in LIPICS. Dagstuhl Publishing, S. 350–364, 2015.

[LTW01] Lepper, Markus; Trancón y Widemann, Baltasar; Wieland, Jacob: Minimze Mark-Up ! – Natural Writing Should Guide the Design of Textual Modeling Frontends. In: Conceptual Modeling — ER2001. Jgg. 2224 in LNCS. Springer, November 2001.

[met] The BandM Metatools Homepage. `http://www.bandm.eu/metatools/`.