Unified Typesetting API – Ein Überblick

Christian Ziesemer

29. November 2005

Dieser Artikel stellt die Unified Typesetting API (UTA) vor, ein Ansatz typografische Dienste zu vereinheitlichen und textverarbeitenden Systemen zur Verfügung zu stellen.

1 Motivation

Seit meinem ersten Kontakt mit Typografie Ende der 90er hat sich mein Wunsch nach einem freien, grafischen DTP-Programm, mit der Satzqualität von TEX nicht erfüllt. Die unterschiedlichen Projekte, die in diese Richtung weiterentwickelt werden könnten, haben wenig bis keinen Fortschritt in den notwendigen Belangen gemacht.

Eine Möglichkeit selbst einzugreifen wäre gewesen, den Mozilla-Browser um entsprechende Funktionen zu erweitern. Mozilla beherrscht inzwischen mathematischen Schriftsatz, bietet allerdings wenig Dokumentation. TEX zu erweitern erschien ebenfalls nicht sinnvoll, da es einem Rolls-Royce gleicht, dem ein Sattelauflieger voller Pakete aufgebürdet wurde. Auch meine Hoffnung, daß in der langen Entwicklungsphase von OpenOffice Fortschritte in typografischer Hinsicht erreicht würden, wurde enttäuscht.

Zwischenzeitlich geben einige Projekte Anlaß zur Hoffnung. Hier sind ε_{χ}TEX, FOP und Scribus zu nennen. Die beiden ersteren werden in ihrer endgültigen Ausbaustufe nahezu identische Aufgaben erfüllen und sich primär im bevorzugten Eingabeformat unterscheiden: TEX vs. XML. Scribus kann als das umfangreichste, freie, grafische DTP-Programm angesehen werden. Die Qualität des Schriftsatzes ist an die QT-Bibliothek gebunden und (noch) nicht für gehobene Ansprüche ausreichend.

2 Ziele

Meist sind neue Anforderungen in der Softwareentwicklung mit großen Umbaumaßnahmen im Produkt verbunden. Das ist der Grund für die Existenz von UTA. Das Projekt hat zum Ziel, diese Umbaumaßnahmen zu verhindern und *allen* textverarbeitenden Systemen typografische Dienste bereitzustellen. Daraus leiten sich folgende Kernforderungen ab:

1. Konzentration auf Schriftsatz und Zeilenumbruch.
2. Unterstützung von multilingualem Schriftsatz.
3. Modularität, um unterschiedliche Qualitätsstufen der Dienste zu erlauben. Dies ist dem Vermeiden von Umbaumaßnahmen geschuldet.
4. Dokumentation des gesammelten/produzierten Wissens.

Abbildung 1 zeigt, wie UTA innerhalb eines kompletten DTP-Systems Anwendung findet.

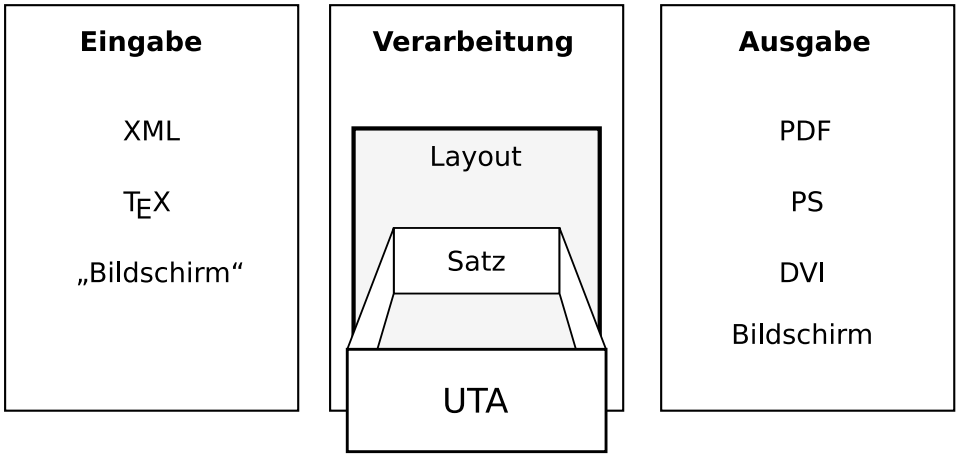

Abbildung 1: Einordnung von UTA in ein Satzsystem.

Anwendungen wie $\varepsilon_{\mathcal{X}}$TEX oder FOP könnten typografische Aufgaben an UTA delegieren und das Ergebnis in das eigene Satzmodell einbauen.

3 Grundlegende Konzepte

UTAs grundlegende Architektur besteht aus drei Schichten, dargestellt in Abbildung 2.

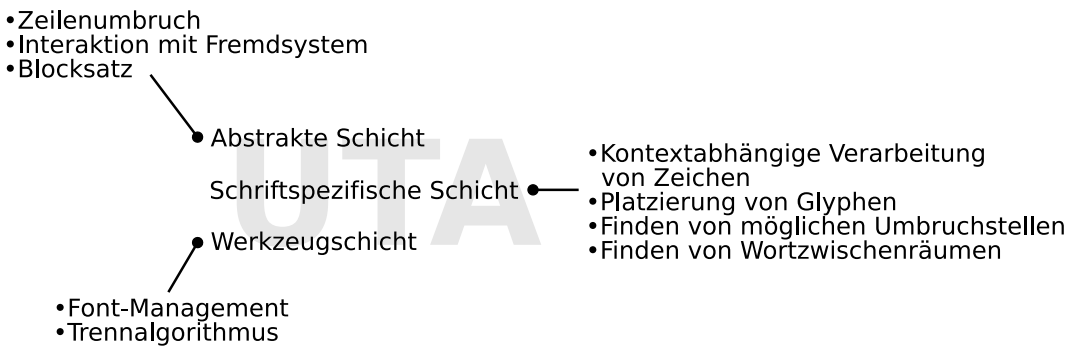

Abbildung 2: Das Schichtenmodell von UTA

Die abstrakte Schicht übernimmt die Kommunikation mit dem Fremdsystem und löst die Probleme Zeilenumbruch und Blocksatz unabhängig von dem verwendeten Schriftsystem. In der schriftsystemspezifischen Schicht kommen Sprachmodule, sog. *Scripts*, zum Einsatz. Sie repräsentieren ein Schriftsystem und kapseln das notwendige Wissen, um Text korrekt zu setzen. Die mittlere Schicht greift auf die Werkzeuge der untersten Schicht zu. Dort sind Trennalgorithmen und Komponenten zum Finden von Schriftarten angesiedelt. Gerade bei der Werkzeugschicht sind Symbioseeffekte angestrebt. Erwähnte Komponenten sind bereits, oder werden in anderen Projekten entwickelt. Bislang ist lediglich die abstrakte Schicht spezifiziert, die anderen stehen noch aus.

Das unterschiedliche Schriftsysteme von entsprechenden Modulen verarbeitet werden, wurde bereits erwähnt. Jedes dieser Scripts setzt Glyphen und eingebettete Grafiken in seiner eigenen „Zeichenfläche“. Die trägt den teils gravierenden Unterschieden in den Schriftsystemen Rechnung. Hier seien exemplarisch nur kontextabhängige Glyphen und die Schreibrichtung genannt.

Die Sprachmodule erzeugen *Items*, die die zentrale Rolle im Zeilenumbruch spielen. Verglichen mit TEX handelt es sich bei einem Item grob um eine beliebige Anzahl von Box und Glue-Objekten. Hinter jedem Item ist ein Umbruch erlaubt, der, wie in TEX, mit einer Strafe belegt werden kann.

Ein Item hat, wie Glue, eine minimale, eine bevorzugte und eine maximale Breite. Zudem ist *vor* dem eigentlichen Umbruch die Position innerhalb der Zeile unbekannt. *Nach* dem Umbruch kann ein Item zu Beginn, in der Mitte, am Ende oder alleine in der Zeile stehen. Ein Umbruchalgorithmus hat damit $3 * 4 = 12$ unterschiedliche Breiten zu bewerten.

Nicht jede Implementierung wird das leisten. Gerade bei einfachen Anwendungen, wie einem Texteingabefeld in einem Formular kommt es nicht auf vollkommenen Umbruch an – auch wenn er nicht schadet. Da es das Ziel von UTA ist, unterschiedlich qualitative Komponenten zu erlauben (vgl. Punkt 3

in obiger Liste), entsteht hier ein völlig neues Problem. Da der Umbruchalgorithmus genauso austauschbar ist, wie die Scripts, kann es zu nicht optimal aufeinander abgestimmten Komponenten kommen. Gibt sich ein Script größte Mühe bei der Berechnung der zwölf möglichen Breiten, aber nur zwei davon werden tatsächlich vom Umbruchalgorithmus unterstützt, ist ein solcher suboptimaler Zustand erreicht. Daher gibt es in UTA eine eigene Komponente, die Informationen über die Fähigkeiten der einzelnen Algorithmen zur Verfügung stellt.

Damit zum letzten Punkt in der Liste der Kernanforderungen, der Dokumentation. Nicht jedes Projekt, das UTA verwenden könnte, wird dies tun. In einem solchen Fall soll zumindest das in UTA gesammelte zur Verfügung stehen. Dokumentation (vor allem einstiegsfreundliche) ist in diesem Bereich leider rar.