TEX, SGML, HTML

SGML – nowy etap w rozwoju TEX-a

Piotr Bolek
i Adam Dawidziuk

Streszczenie

Artykuł jest próbą pokazania praktycznych zastosowań standardu SGML. Z naszych doświadczeń wynika, że stosowanie tego standardu w pracy redakcji i wydawnictwa jest dobrym uzupełnieniem systemu TEX. Wprowadzenie dodatkowego etapu w przetwarzaniu tekstu powoduje w tym wypadku znaczne obciążenie komputera, ale współczesne, średniej klasy, komputery PC dają sobie radę. Poza elastycznością formatu źródłowego i możliwością tworzenia wielu form ostatecznych publikacji, umiejętne stosowanie języka SGML pozwala na automatyczne wykonywanie wielu skomplikowanych działań na tekście.

Niniejszy artykuł powstał w oparciu o treść referatu prezentowanego przez nas na konferencji BachoTEX'97.

Wprowadzenie

Do rzeczy dobrych przyzwyczajamy się łatwo. Kto dziś pamięta jak wyglądały komputery w latach 70-tych? Warto uprzytomnić sobie, że komputer osobisty pojawił się w latach 80-tych, przy czym na samym początku możliwości tych „komputerków" były niewielkie, w przeciwieństwie do ceny. Powszechną, w miarę niedrogą maszyną, która mogła być wykorzystana do pewnych zadań praktycznych był komputer PC z procesorem 8086, bardzo popularny w Polsce w końcu lat 80-tych.

Gwałtowny rozwój mikroelektroniki umożliwił stworzenie procesorów kolejnych generacji. Moc obliczeniowa komputera stojącego na biurku przekroczyła możliwości ogromnych maszyn stacjonarnych. Okazało się, że komputer osobisty z procesorem 80386 liczy tak samo szybko jak potężna maszyna obsługująca ośrodek obliczeniowy dużej instytucji, a do przewożenia dysku o pojemności 20 GB nie jest koniecznie potrzebny wózek akumulatorowy. Współczesny niedrogi komputer PC, który często służy wyłącznie do zabawy, ma olbrzymią moc obliczeniową i jeszcze kilka lat temu nosiłby nazwę „superkomputera".

Nawet jeśli używamy bardzo rozbudowanych makr do TEX-a (np. LATEX-a) i wszystkie fonty generujemy w locie, to po dokładnym sprawdzeniu okaże się, że przez większą część czasu pracy nasz komputer nie ma nic do zrobienia. Po prostu się nudzi.

Jednocześnie, wraz z rozwojem różnych technik prezentacji okazuje się, że nasza praca z publikacją nie ogranicza się wyłącznie do przygotowania materiałów do druku. Coraz częściej jesteśmy zmuszani do tworzenia wersji elektronicznej publikacji. Oznacza to sporo dodatkowej pracy. Przy czym, przy wznowieniu okazuje się, że trzeba tekst opracować na nowo, bo obowiązuje nowa wersja języka HTML, a jeszcze na dokładkę mamy wprowadzić dokładniejszy skorowidz do bazy danych i stworzyć wersję w PDF-ie, w której odnośniki i adresy będą aktywne. Można się zapracować na śmierć – ale chwileczkę – my gonimy resztką sił, a na naszym biurku stoi nasz nowy komputer, który „nic" nie robi.

I w ten sposób dochodzimy do wniosku, że TEX – owszem, ale często potrzeba jakby czegoś więcej.

Etapy powstawania publikacji

Kiedy już uświadomimy sobie, że w dzisiejszym trudnym świecie nie jest łatwo poprzestać wyłącznie na używaniu TEX-a, to należy zastanowić się, co i jak dałoby się usprawnić. Proces powstawania publikacji można podzielić na kilka etapów. Tradycyjnie, z pominięciem korekty, wyglądały one mniej więcej tak:

1. opracowanie rękopisu – autor przy pomocy słów starał się wyjaśnić, co ma na myśli;
2. opracowanie typograficzne – redaktorzy starali się ustalić strukturę przekazywanej informacji i podkreślić ją stosowną formą graficzną;
3. skład i druk.

Do realizacji każdego z tych etapów potrzebna była odmienna wiedza i różne narzędzia, w związku z tym podział kompetencji (a więc i odpowiedzialności) był wyraźny. Gdy pojawiły programy komputerowe umożliwiające robienie składu na biurku, „naturalny" porządek rzeczy uległ zakłóceniu i autor sam zajął się składem. Konieczność uwzględniania prezentacji podczas pisania tekstu nie ułatwia autorowi pracy. Jest to szczególnie dotkliwe przy systemach typu WYSYWIG, ale nawet użytkownik TeX-a często zastanawia się, co „powiedzieć" kursywą, a co nie.

Poszukując usprawnienia pracy nad publikacją, można próbować oddzielić pracę autora od nadawania publikacji konkretnego wyglądu. Takie podejście wydaje się mieć sens w dobie publikacji elektronicznych, kiedy tak naprawdę nie wiemy, jaka będzie ostateczna postać tekstu. Wysiłki autora (i redakcji merytorycznej) należałoby ukierunkować na dokładny opis struktury publikacji. Autor nie powinien się zastanawiać, czy w tekście naukowym nazwy wprowadzanych nowych pojęć należy pogrubić, czy nie, ale powinien mieć możliwość oznaczania, co jest takim nowym terminem. Do tego celu dość dobrze nadaje się język oznaczania struktury tekstu SGML (ang. *Standard Generalized Markup Language*).[1]

Jeśli uwzględnimy współczesne realia i do wymienionych etapów tworzenia publikacji dołączymy jej „strukturalizację" na samym wstępie, to etapy przygotowania publikacji w „epoce informatycznej" będą wyglądały następująco:

1. autor i redaktor merytoryczny opracowują tekst z naniesioną strukturą logiczną (np. SGML);
2. typograf (grafik) określa sposób prezentacji poszczególnych elementów logicznych tekstu, a odpowiednio sformatowany tekst (np. TeX, HTML, PostScript, PDF) jest tworzony przez oprogramowanie;
3. sformatowany tekst jest prezentowany na odpowiednim urządzeniu wyjściowym (np. naświetlarka, ekran monitora).

Można mówić o kolejnych „warstwach przetwarzania" publikacji: strukturalnej (SGML), formatowania wysokiego poziomu (TeX, HTML), formatowania niskiego poziomu (PostScript, PCL) i warstwie prezentacyjnej. Warto zwrócić uwagę, że automatyczne przetwarzanie publikacji jest łatwe tylko w jedną stronę: od struktury do prezentacji. Przetwarzanie w drugą stronę wymaga interpretowania treści przekazu i, jak dotąd, zdecydowanie przekracza możliwości komputera. Tekst z oznaczoną strukturą logiczną jest więc w naszym schemacie najbardziej ogólną i otwartą formą opisu publikacji, z której można otrzymać „wszystko".

Opis struktury tekstu

Dlaczego warto? Opisywanie struktury tekstu przy pomocy znaczników jest wprowadzaniem dodatkowych informacji do tekstu. Jest to „informacja o informacji" – przy pomocy znaczników określamy czym jest zawartość pewnych pól. Pozornie oznacza to mnóstwo dodatkowej pracy i to już na samym początku procesu przygotowania publikacji. Praca ta w zasadzie spada na autora, bo tylko on tak naprawdę wie (lub powinien wiedzieć) „co jest czym" w jego tekście. Korzyści z tego wynikające są liczne, więc wymienimy tylko najważniejsze z nich.

Uproszczenie pracy redakcyjnej. Jeśli autor umieści w tekście opis struktury, to właściwie redakcja techniczna przestaje być potrzebna, a tak naprawdę zmieniają się jej funkcje. Tradycyjnie i tak autor określał znaczenie poszczególnych słów w rękopisie, wskazując co jest tytułem, co podtytułem itd. Redaktor merytoryczny kontrolował, czy ma to sens, a redaktor techniczny nadawał strukturze formę.

W tekście strukturalnym elementy tekstu są oznaczone, a redakcja techniczna tworzy program, który automatycznie nada strukturze odpowiednią formę. Znaczniki są np. tłumaczone na makra TeX-owe – wystarczy stworzyć styl, czyli zdefiniować znaczenie tych makr. Unika się w ten sposób żmudnego i czasochłonnego etapu wprowadzania odpowiednich makr do tekstu.

Unikanie błędów strukturalnych. Pracując z tekstami strukturalnymi nie da się popełnić wielu błędów, o które łatwo w metodach tradycyjnych. Skoro wszystkie elementy tekstu są oznaczone, to można podać dopuszczalne zależności między tymi elementami (DTD[2]) i sprawdzać poprawność dokumentu. Na przykład w TeX-u ktoś może napisać:

```
\begin{abstract}
 Tu streszczenie...
\end{abstract}
```

1: Nie sugerujemy, że standard SGML jest ze swej natury niezależny od prezentacji – można go wykorzystać w ten sposób, żeby autor nie musiał wiedzieć, co to jest kursywa.

2: *Document Type Definition* – definicja typu dokumentu (przyp. red.).

```
\subsection{Tytuł pierwszego rozdziału}
Treść pierwszego rozdziału...
\subsection{Tytuł pierwszego
 podrozdziału}
Treść pierwszego podrozdziału...
\section{Tytuł drugiego rozdziału}
...
```

I rezultat będzie dość nieoczekiwany: Tytuł pierwszego rozdziału będzie wyglądał jak tytuł podrozdziału i jeśli numerujemy podrozdziały, to zobaczymy numer 0.1. Wbrew pozorom nie wszystkie błędy tego rodzaju łatwo wyłapać podczas korekty. Znane są przypadki poważnych publikacji z literówką w tytule. Bywają też błędy mniej rzucające się w oczy.

Problem tego typu zupełnie nie ma szans zaistnieć w przypadku tekstu strukturalnego i stosowania parsera lub edytora strukturalnego z wbudowanym parserem. Podczas analizy składni zostanie zgłoszony komunikat błędu. Przy stosowaniu edytora strukturalnego powyższego fragmentu nie uda nam się wpisać źle. Będzie on wyglądał na przykład tak:

```
⟨abstract⟩
Tu streszczenie...
⟨/abstract⟩
⟨sect1⟩
⟨title⟩Tytuł pierwszego rozdziału⟨/title⟩
⟨para⟩Treść pierwszego rozdziału...
⟨/para⟩
⟨sect2⟩
⟨title⟩Tytuł pierwszego podrozdziału
⟨/title⟩
⟨para⟩Treść pierwszego podrozdziału...
⟨/para⟩
⟨/sect2⟩
⟨/sect1⟩
⟨sect1⟩
⟨title⟩Tytuł drugiego rozdziału⟨/title⟩
⟨para⟩Treść drugiego rozdziału...⟨/para⟩
⟨/sect1⟩
```

Łatwość wprowadzania tekstu. Dokument strukturalny z powyższego przykładu raczej nie wygląda przyjaźnie. Odstrasza liczba znaczników. Na szczęście nie trzeba ich wprowadzać ręcznie. Potrafi to robić edytor. Warto zwrócić uwagę, że dokument składa się z „pudełek w pudełkach". Rozdział, czyli pudełko ⟨*sect1*⟩ ma zaznaczony początek i koniec. Podrozdział, czyli pudełko ⟨*sect2*⟩ musi całkowicie zawierać się w pudełku ⟨*sect1*⟩. Informacja o tym jakie pudełka mogą znajdować się w jakich pudełkach zapisana jest w DTD. Edytor czyta DTD i na bieżąco sprawdza w jakim elemencie tekstu znajduje się kursor. Dzięki temu wystarczy podać polecenie „wstaw element" i wskazać jeden z elementów dopuszczalnych w danym miejscu (np. podając pierwszą literę nazwy elementu), aby na ekranie pojawiły się odpowiednie znaczniki. Jeśli wstawiane pudełko musi zawierać określone znaczniki (np. dla rozdziału obowiązkowym elementem jest tytuł) – edytor zapyta o zawartość tych pól.

Stosunkowo prosty i jednolity format wprowadzania znaczników w tekst pozwala tworzyć edytory inteligentnie interpretujące strukturę dokumentu. Znaczniki wcale nie muszą być widoczne na ekranie, a budowa poprawnego strukturalnie dokumentu może być wspierana na wiele sposobów. Podlegający nieustannej i dyskretnej kontroli autor może skupić się na treści przekazu i zupełnie nic nie musi wiedzieć o wyglądzie publikacji, ani tym bardziej o typografii. A o to przecież nam chodziło.

TeX jako język strukturalny. Skoro koncepcja formalnej strukturalizacji tekstu sprowadza się do wstawiania w odpowiednie miejsca wcześniej zdefiniowanych znaczników, to dlaczego nie używać TeX-a jako języka opisu dokumentów strukturalnych? Przede wszystkim dlatego, że należałoby znacznie ograniczyć możliwości TeX-a. W TeX-u można przecież pisać zupełnie niestrukturalnie. Zamiast definiować makro `\title{}` możemy zmienić w tekście czcionkę na większą i wprowadzić stosowne odstępy pionowe. Każdy, kto był pracownikiem redakcji technicznej ma zwykle sporo do powiedzenia na temat autorów piszących w taki sposób.

Jako język strukturalny dużo lepiej wypada LaTeX. Nawet nie mający doświadczenia w składzie autor może stworzyć w LaTeX-u dokument, który po dołożeniu odpowiedniego stylu będzie wyglądał bardzo dobrze.

TeX jest więc świetny, ale zbyt ogólny. Elementy w tekście możemy oznaczać na wiele sposobów i już stworzenie edytora, który nadzorowałby oznaczanie tekstu nie jest łatwe. Przy pewnych założeniach można by pisać w TeX-u i kontrolować automatycznie poprawność struktury tekstu. Byłby to jednak nie TeX, ale jego bardzo ograniczony podzbiór. Poza tym nie ma potrzeby wynajdowania koła na nowo. Taki „podzbiór" TeX-a został już opracowany (SGML), a nawet określony normami (ISO 8879). Istnieją odpowiednie narzędzia. Poza tym, jeżeli chcemy używać koncepcji znaczników, to na dobrą sprawę jest nam wszystko jedno, jak te znaczniki wprowadzamy. Formalnie nie ma różnicy między ⟨*emphasis*⟩wyróżnienie⟨/*emphasis*⟩, a `\emph{wyróżnienie}`.

Nowe miejsce dla TEX-a

Wniosek, że TEX nie będzie potrzebny jest zbyt pochopny. Tekst z oznaczoną strukturą musi zostać przetworzony do formatu prezentacyjnego. Skład i łamanie tekstu nie jest sprawą prostą, a TEX to potrafi. Nie ma więc najmniejszego powodu rezygnowania z TEX-a.

Usprawnienie polega głównie na nowej metodzie przygotowywania danych wejściowych do TEX-a. Używanie TEX-a sprowadza się do stworzenia zestawu makr, czyli stylu, określającego wygląd dokumentu. Potem trzeba tylko określić na jakie makra przetłumaczyć poszczególne znaczniki. Z TEX-a zostaje więc to co najprzyjemniejsze – znając strukturę tekstu możemy skoncentrować się na zrealizowaniu odpowiedniej do treści formy graficznej.

Powszechnie znana jest bezkonkurencyjność TEX-a w dziedzinie składu wzorów matematycznych. W specyfikacji standardu SGML nie określono sposobu wprowadzania wzorów matematycznych. Można więc śmiało korzystać z dobrodziejstw TEX-a, który przecież jest standardem stosowanym przez wszystkie poważne wydawnictwa matematyczne.

Zastosowania praktyczne

Powyższe rozważania mają charakter bardzo teoretyczny. Słuszność prezentowanych tutaj poglądów sprawdziliśmy w praktyce. Przez ostatnie dwa lata wszystkie pisane przez nas publikacje tworzyliśmy wykorzystując SGML.

Różnica w stosunku do bezpośredniego przygotowywania danych wejściowych dla TEX-a jest zauważalna. Bardzo wygodna jest praca z „inteligentnym” edytorem. Używany przez nas emacs wydaje się nam bardziej przyjazny w trybie SGML niż ten sam emacs wspomagający tworzenie dokumentu TEXowego. Nie bez znaczenia jest możliwość łatwego poruszania się w tekście wg. znaczników, możliwość chowania wybranych elementów itd.

Oznaczanie elementów tekstu jest łatwe i można sobie pozwolić na wprowadzanie znaczników w wielu różnych miejscach. W wielu przypadkach elementy te w żaden sposób nie będą wyróżnione w tekście, ale znakomicie upraszczają np. tworzenie skorowidzów, czy odnośników.

Z dużymi tekstami można pracować jak z bazą danych. Jest to przydatne nie tylko do opracowania wypisów i wersji hipertekstowych dokumentu, ale także do wyszukiwania błędów podczas korekty. Ostatnio składany przez nas tekst książki miał objętość prawie 60 arkuszy wydawniczych. Przy użyciu prostych skryptów napisanych w języku perl możliwe było np. uzyskanie alfabetycznego spisu wszystkich nazw programów (książka była o systemie operacyjnym) pojawiających się w tekście. W ten sposób znaleźliśmy kilkaset błędów, które nie zostały zauważone podczas trzech korekt (strach pomyśleć ile błędów zostało).

Miłą cechą stosowania standardu SGML okazała się możliwość uzależnienia tłumaczenia znaczników na makra TEX-owe w zależności od kontekstu. Element będący nazwą programu inaczej wyglądał w tekście, inaczej w tabeli, inaczej w tytule, inaczej w notce, a jeszcze inaczej w skorowidzu. Napisanie w TEX-u makra zależnego od kontekstu nie zawsze jest możliwe, bo nie zawsze kontekst jest wyraźnie określony – a na pewno nie jest to zadanie trywialne. Przy wykorzystaniu SGML było to zadanie nieskomplikowane. Jako autorzy mogliśmy zawsze używać tego samego znacznika w każdym kontekście.

Podobne doświadczenia mieliśmy pracując z dużą grupą autorów. Jako standard wymiany dokumentów przyjęliśmy HTML, którego elementom nadaliśmy stosowne znaczenia. Błędy struktury wykrywał parser, a nam pozostało niewiele do zrobienia.

Trudno nam oszacować ile tak naprawdę zaoszczędziliśmy czasu używając standardu SGML przez ostatnie dwa lata. Na pewno wprowadzanie tekstu (praca autora) nie trwa dłużej niż pisanie w TEX-u. Natomiast liczba godzin spędzonych przy składzie zmniejszyła się kilkakrotnie.

Z naszych doświadczeń wynika również, że automatyczny skład nie zawsze wychodzi. Dokładnie tak samo jak przy składzie w TEX-u – jeśli zależy nam na wyglądzie publikacji, to czasem musimy „ręcznie” złamać stronę lub podpowiedzieć jak łamać wyraz (czy wręcz zmienić fragment tekstu). Wystarczy do tego kilka prostych znaczników w SGML, wskazujących miejsca łamania.

W naszej pracy z SGML wykorzystujemy różnego rodzaju oprogramowanie. Przede wszystkim edytor. Podstawowym używanym przez nas edytorem, który potrafi działać jako edytor strukturalny jest emacs. Emacs jest programem, który ma wiele twarzy – jedną z nich może być „twarz strukturalna” – działa on jako sgmlowy edytor strukturalny w trybie PSGML[3]. Tryb ten zawiera prosty parser sgmlowy i potrafi pracować z dowolnym DTD. My najczęściej używamy DTD DocBook w wersji 3.0.[4] opracowany przez Davenport Group[5]. Przy użyciu zestawu emacs+PSGML+DTD można także bardzo wygodnie pisać w HTMLu, co też nam się czasem zdarza.

Kolejnym ważnym narzędziem w pracy z SGML jest parser. Służy on do weryfikacji zgodności dokumentu z DTD. Wykorzystywany jest także przy konwersji dokumentów do innych formatów. Najlepszym dostępnym obecnie parserem jest nsgmls (z pakietu SP) autorstwa Jamesa Clarka[6]. Parser ten jest dostępny nieodpłatnie.

Przy pracy z dokumentami używamy także perla. Jest on już standardowym narzędziem dostepnym w każdym porządnym systemie. Do przetwarzania dokumentów sgmlowych na inne formaty wykorzystujemy napisaną przez Davida Megginsona obiektową bibliotekę perlową SGMLSpm oraz wykorzystujący ją program perlowy sgmlspl.[7] Narzędzia te umożliwiają przetwarzanie wyjścia z parsera nsgmls. Na potrzeby konwersji danego DTD do jakiegoś formatu prezentacyjnego trzeba napisać prosty skrypt, opisujący akcje (najczęściej wypisanie na wyjście odpowiednich poleceń formatujących w danym języku) związane z pojawiającymi się elementami strukturalnymi.

Podsumowanie

Zastosowanie standardu SGML może znacznie zmniejszyć pracochłonność opracowania publikacji. Wprowadzanie tekstu w formacie niezależnym od prezentacji jest znacznym ułatwieniem dla autora. Do przygotowania artykułu zwykle wystarcza ten sam opis struktury dokumentu (DTD) bez względu na to do jakiego czasopisma kierowana jest praca. W efekcie, autor na ekranie widzi zawsze to samo i może koncentrować się na zagadnieniach merytorycznych. Ważne jest używanie dobrego edytora strukturalnego, który „pilnuje” formalnej poprawności dokumentu i usprawnia pracę.

Dobrze przygotowany tekst strukturalny nie sprawia wiele kłopotu w redakcji. Można go przetworzyć do dowolnej postaci. Ostateczny wygląd dokumentu można zmieniać zależnie od potrzeb.

Istotnym aspektem zastosowania standardu SGML jest możliwość całkowitego odseparowania autora od wyglądu publikacji. Autor nie musi uczyć się nawet podstaw typografii, ale jednocześnie wykonuje „czarną robotę” składu, czyli wprowadza znaczniki do tekstu. Kto więc robi skład? Ano, komputer (może do spółki z autorem), a o to nam przecież od początku chodziło.

TEX łącznie z wiedzą tajemną z nim związaną ma w całym procesie swoje własne i ważne miejsce, lecz pozostaje niedostępny dla rzesz profanów, którym wydaje się, że doktorat z nauk ścisłych upoważnia do negowania doświadczeń pokoleń zecerów i typografów.

Trudno jednak wróżyć z fusów i prorokować, że już niebawem powstanie dostępne dla każdego oprogramowanie pozwalające korzystać szerokim rzeszom z dobrodziejstw współczesnej informatyki. Jak wygląda rzeczywistość rynku oprogramowania, każdy widzi, ale to zupełnie inna historia. Póki co nie bardzo wiemy, jak zachęcić autorów do pisania w SGML.

3: ftp.lysator.liu.se/pub/sgml/

4: ftp.ora.com/pub/davenport/docbook/

5: www.ora.com/davenport/index.html

6: www.jclark.com/sp/index.htm

7: Dowolne lusterko archiwum perlowego CPAN – np. sunsite.icm.edu.pl/pub/CPAN

Literatura

[1] Charles F. Goldfarb, *The SGML Handbook*, Clarendon Press, Oxford, 1995.

[2] Martin Bryan, *SGML An Author's Guide to the Standard Generalized Markup Language*, Addison-Wesley, 1995.

[3] Larry Wall, Tom Christiansen, Randal L. Schwartz, *Programming Perl*, O'Reilly, 1996.

◇ Piotr Bolek
P.Bolek@ia.pw.edu.pl

◇ Adam Dawidziuk
A.Dawidziuk@elka.pw.edu.pl