Zpravodaj Československého sdružení uživatelů TeXu

Zdeněk Wagner

Sazba obrazové publikace s plovoucím textem

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 23 (2013), No. 3-4, 133–156

Persistent URL: http://dml.cz/dmlcz/150203

Sazba obrazové publikace s plovoucím textem

Zdeněk Wagner

Algoritmy TEXu jsou určeny pro zpracování textu, ale nezabývají se prací s externími obrázky. Tato činnost je ponechána výstupním ovladačům. Ovladače se přizpůsobily vývoji polygrafie, takže zejména s využitím balíčku GRAPHICX lze vkládat obrázky v běžných formátech pohodlným způsobem nezávislým na použitém ovladači. Formát LATEX je primárně určen pro sazbu textů, v nichž se občas vyskytuje tabulka či obrázek. Mechanismus plovoucích objektů zajistí, že si tabulky a obrázky automaticky najdou vhodné místo v textu. Jiná situace však nastane v případě, kdy jsou těžištěm publikace právě obrázky, jejichž umístění na stránkách je pevně dáno a jež má text obeplout. Navíc má text uprostřed odstavce přetéci na další stránku, kde bude sázen do sloupce jiné šířky. V tomto okamžiku již nelze použít existující postupy, je nutno napsat vlastní makra. V tomto článku je popsáno, jak je využito spojení XML s XELATEXem a vlastními makry při sazbě ilustrované publikace.

Klíčová slova

plovoucí text, XELATEX, XML, XSLT, validace

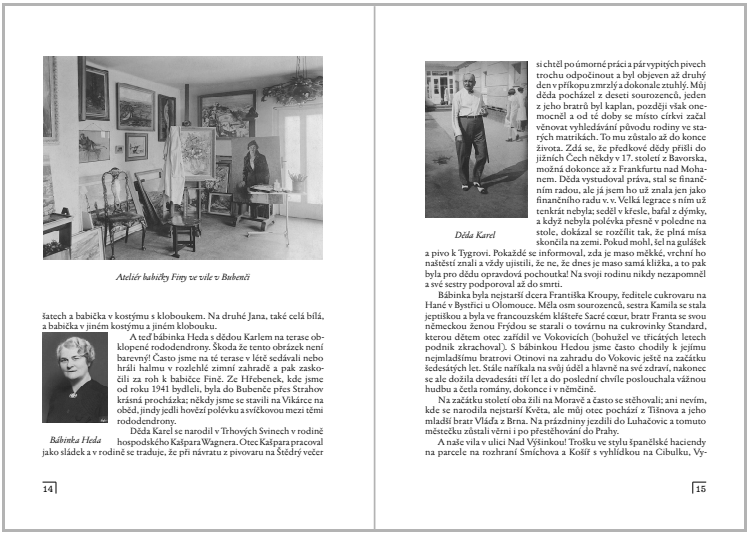
1. Typy obrazových publikací

Publikace, ať už odborné, nebo beletrie, obsahují nejen text, ale též ilustrace, které text doprovázejí. Mohou to být jak diagramy a grafy, tak fotografie. Do textu mohou být začleněny různými způsoby a podle toho klasifikujeme typy obrazových publikací.

LATEX byl vytvořen zejména pro zpracování matematických textů. Zpracování rovnic a tabulek nečiní problémy. Počítá se s tím, že text je doprovázen poměrně malým počtem plovoucích objektů, tabulek a obrázků, které zabírají celou šířku tiskového sloupce, případně celou šířku tiskového zrcadla. V textu mohou být i celostránkové plovoucí objekty. Důležité je přitom pouze zachování vzájemného pořadí tabulek a obrázků, na jejich přesném umístění nezáleží. Standardní plovoucí prostředí tyto případy snadno řeší. Zcela bezproblémové jsou i velké zásahy do textu při korekturách, protože LATEX najde plovoucím objektům jiná, vhodná místa. Příkazy pro pevné ukotvení plovoucích objektů jsou též dostupné, ale ty používáme až v závěrečné fázi, pokud s automatickým umístěním nejsme spokojeni.

V odborné literatuře i v beletrii se však setkáváme s případy, kdy obrázek vyžaduje pouze malou část šířky sazby. Použití klasického LATEXového přístupu

doi: 10.5300/2013-3-4/133



Ateliér babičky Finy ve vile v Bubenči

šatech a babička v kostýmu s kloboukem. Na druhé Jana, také celá bílá, a babička v jiném kostýmu a jiném klobouku.

A teď bábinka Heda s dědou Karlem na terase obklopené rododendrony. Škoda že tento obrázek není barevný! Často jsme na té terase v létě sedávali nebo hráli halmu v rozlehlé zimní zahradě a pak zaskočili za roh k babičce Fině. Že Hřebenek, kde jsme od roku 1941 bydleli, byla do Bubenče přes Strahov krásná procházka; někdy jsme se stavili na Vikárce na oběd, jindy jedli hovězí polévku a svíčkovou mezi těmi rododendrony.

Bábinka Heda

Děda Karel se narodil v Trhových Svinech v rodině hospodského Kašpara Wagnera. Otec Kašpar pracoval jako sládek a v rodině se traduje, že při návratu z pivovaru na Štědrý večer

Děda Karel

si chtěl po úmorné práci a pár vypitých pivech trochu odpočinout a byl objeven až druhý den v příkopu zmrzlý a dokonale ztuhlý. Můj děda pocházel z deseti sourozenců, jeden z jeho bratrů byl kaplan, později však onemocněl a od té doby se místo církvi začal věnovat vyhledávání původu rodiny ve starých matrikách. To mu zůstalo až do konce života. Zdá se, že předkové dědy přišli do jižních Čech někdy v 17. století z Bavorska, možná dokonce až z Frankfurtu nad Mohanem. Děda vystudoval práva, stal se finančním radou, ale já jsem ho už znala jen jako finančního radu v. v. Velká legrace s ním už tenkrát nebyla; seděl v křesle, bafal z dýmky, a když nebyla polévka přesně v poledne na stole, dokázal se rozčílit tak, že plná mísa skončila na zemi. Pokud mohl, šel na guláš a pivo k Tygrovi. Pokaždé se informoval, zda je maso měkké, vrchní ho naštěstí znali a vždy ujistili, že ne, že dnes je maso samá kližka, a to pak byla pro dědu opravdová pochoutka! Na svoji rodinu nikdy nezapomněl a své sestry podporoval až do smrti.

Bábinka byla nejstarší dcera Františka Kroupy, ředitele cukrovaru na Hané v Bystřici u Olomouce. Měla osm sourozenců, sestra Kamila se stala jeptiškou a byla ve francouzském klášteře Sacré cœur, bratr Franta se svou německou ženou Frýdou se starali o továrnu na cukrovinky Standard, kterou dětem otec zařídil ve Vokovicích (bohužel ve třicátých letech podnik zkrachoval). S bábinkou Hedou jsme často chodily k jejímu nejmladšímu bratrovi Otinovi na zahradu do Vokovic ještě na začátku šedesátých let. Stále nařídkala na svůj úděl a hlavně na své zdraví, nakonec se ale dožila devadesáti tří let a do poslední chvíle poslouchala vážnou hudbu a četla romány, dokonce i v němčině.

Na začátku století oba žili na Moravě a často se stěhovali; ani nevím, kde se narodila nejstarší Květa, ale můj otec pocházel z Tišnova a jeho mladší bratr Vlasta z Brna. Na prázdniny jezdili do Luhačovic a tomuto městečku zůstali věrni i po přestěhování do Prahy.

A naše vila v ulici Nad Výšinkou! Trošku ve stylu španělské haciendy na parcele na rozhraní Smíchova a Košíř s vyhlídkou na Cibulku, Vy-

Obrázek 1: Ukázka sazby kombinující neobtékané a obtékané obrázky

by bylo dosti nevhodné. Výhodnější je, když je obrázek obtékán textem. Oba přístupy lze kombinovat, jak vidíme na dvojstraně z knihy *Úniky z temna* [14] na obrázku 1. Výhodou obtékaných obrázků je kompaktní vzhled a úspora místa. Nevýhodou je složitější sazba. Obtékaný obrázek je na rozdíl od plovoucího pevně zakotven ke konkrétnímu odstavci. Pokud při korektuře provedeme větší zásah do textu, může se posunout stránkový zlom do takového místa, že se část obtékaného obrázku dostane mimo sazbu. Navíc se může porušit synchronizace mezi obtékanými a plovoucími obrázky.

Na druhé straně spektra stojí publikace obrazové, které neobsahují text. Může tak být sestavena celá kniha, nebo jen obrazová příloha. Obrázky mohou být opatřeny krátkými popisky, které se vždy vztahují ke konkrétnímu obrázku či skupině obrázků. Jsou tedy na stejné straně s obrázky. Sazba proto nečiní zvláštní potíže, používáme vizuální značkování. Problém nastane pouze tehdy, když je při korektuře rozhodnuto, že se má nějaký obrázek vypustit nebo přidat. Pak se obvykle značná část sazby rozpadne a velkou část knihy je nutno přesadit.

Publikace s plovoucím textem leží uprostřed mezi výše zmíněnými typy. Obrázek může mít krátký popisek. Současně však publikace obsahuje dlouhý, souvislý text. Na rozdíl od prvního typu, kdy obrázky doplňují text, je zde situace opačná. Těžištěm je obrazová informace, kterou text doplňuje. Umisťování obrázků na stránky tedy nelze svěřit automatu, protože je podřízeno výtvarnému záměru. Standardní nástroje LaTeXu dovedou sázet text, do něhož jsou vkládány plovoucí

objekty. Zde však potřebujeme pevně ukotvit obrázky a text musí proplouvat volnými oblastmi, ale nesmí proplout k obrázkům z následující kapitoly. Protože text není pevně spjat s konkrétním obrázkem, budeme vyžadovat stránkové zlomy uprostřed odstavců jako při běžné sazbě. Obrázky budou často obtékané, takže po stránkovém zlomu pokračuje sazba odstavce ve sloupci jiné šířky. Je zřejmé, že LaTeX pro řešení tohoto problému žádné hotové řešení nenabízí. Je tedy nutné vytvořit vlastní makra.

Softwarový systém, který bude popsán v následujícím textu, byl vytvořen pro knihu *Barvy Indie* [12]. Až dodatečně jsem získal informaci [2], že stejný problém řešil Petr Olšák. Své řešení však nezveřejnil v otevřené literatuře ani na svých webových stránkách [17]. V době získání této informace sice kniha ještě nebyla dokončena, ale systém pro její sazbu byl již hotov a bylo napsáno několik kapitol. Z tohoto důvodu jsem se nesnažil řešení Petra Olšáka získat.

2. Požadavky na sazbu publikace s plovoucím textem

Z předchozího rozboru plyne, jaké požadavky budeme na sázecí systém klást. Z obecného hlediska musí software umět:

1. ukotvit obrázek na pevně dané místo na stránce;
2. připojit popisek obrázku (může být víceřádkový);
3. zobrazit rozložení obrazových a textových oblastí;
4. zadat text bez určení stránky, kde bude zobrazen;
5. vložit text na volnou část stránky (textová oblast);
6. povolit stránkový zlom uprostřed odstavce;
7. umožnit sazbu do sloupce jiné šířky po stránkovém zlomu;
8. neodsunout text k obrázkům z následující kapitoly;
9. dodržet řádkový rejstřík.

Kromě toho je nutné mít na zřeteli, že systém je vyvíjen pro zcela konkrétní knihu [12], která je psána česky, ale vlastní jména a geografické názvy jsou uváděny též hindsky, v některých případech i urdsky. Hindština používá dévanágarské písmo, urdština arabské písmo, píše se tedy zprava doleva. Potřebujeme proto podporu těchto písem.

Při hledání vhodného systému si musíme odpovědět na několik otázek, jež jsou sice do jisté míry samostatné, ale přesto vzájemně svázané. Pokud bychom na první z následujících otázek odpověděli jinak, neměly by další otázky smysl a museli bychom je nahradit jinými.

2.1. Volba typografického systému

Jak jsme si již vysvětlili, LaTeX nenabízí hotový balíček, který by splnil všechny požadavky. Můžeme se tedy rozhlédnout, zda tyto funkce nenabízí jiný nástroj.

V kapitole 2.4 si ukážeme, že zápis zdrojového textu v XML nabízí užitečné možnosti. Nebylo by tedy vhodné použít čistě nástroje XML? Jazyk XML definuje strukturu souboru, ale nikoliv jeho vzhled. Pro tisk je nutno použít další nástroj, například formátovací objekty, XSL-FO. Rozšířené požadavky na formátovací objekty již byly navrženy [1] a je velmi pravděpodobné, že s jejich pomocí by se dala kniha s plovoucím textem vysázet. Návrh však ještě neznamená, že jsou tyto vlastnosti ve všech procesorech formátovacích objektů implementovány. Pokud bychom kupovali komerční procesor jen kvůli sazbě jedné knihy, pak by překvapení způsobené tím, že verze 2.0 implementována není, bylo dosti nemilé.

Zajímavou alternativou by mohl být passiveTEX [4]. Zde však není implementován ani kompletní standard 1.0, takže se lze oprávněně obávat, že s jeho pomocí knihu s plovoucím textem nevysázíme.

Víme však, že TEX za jistých okolností požadovanou úlohu zvládne. Zvolíme tedy TEX jako základní nástroj a budeme hledat pomůcky, které nám práci při psaní textu a při sazbě usnadní.

2.2. Volba sázecího stroje (enginu)

Anglická terminologie distribuce TEX Live používá výraz *engine*, pro nějž jsem nenašel jiný český ekvivalent než *sázecí stroj*. Myslí se tím konkrétní spustitelný program implementující algoritmy TEXu. K dispozici máme tři různé sázecí stroje.

Prvním strojem je původní osmibitový TEX. Z hlediska podpory makrojazyka obsahuje vše, ale my potřebujeme sázet v jazycích, které používají nelatinková písma. To je však v osmibitovém TEXu komplikované. Zdrojový text je nutno psát pomocí latinkové transliterace, v případě dévanágarského písma musíme použít preprocesor. Tento sázecí stroj je tedy použitelný, ale není pohodlný.

LuaTEX nabízí dvě nové možnosti, práci v šestnáctibitovém kódování Unicode a skriptovací jazyk lua. Na rozdíl od latinky však arabské a dévanágarské písmo představuje problém. Při spojování písmen je nutno aplikovat řadu pravidel. Tato pravidla jsou implementována přímo v jazyce lua. Zdaleka ale ještě nejsou vytvořena pro všechna písma. Zatímco podpora arabského písma je na vysoké úrovni, implementace indických písem odvozených z písma bráhmí zcela chybí. LuaTEX je tedy pro sazbu hindských textů zcela nepoužitelný.

XETEX sice nemá skriptovací jazyk, ale při práci s písmem využívá standardní knihovnu. Do verze distribouvané s TEX Live 2012 to byla knihovna ICU, nyní je používána knihovna Harfbuzz. Použití indických písem proto nepředstavuje žádný problém. Pro sazbu české knihy obsahující texty v hindštině a urdštině je tedy XETEX ideálním řešením.

2.3. Volba formátu

Formát je v podstatě sada maker, která byla při inicializaci zapsána v binární podobě primitivem `\dump`, a při startu sázecího stroje jsou definice rychle načteny

do paměti. Tato makra máme k dispozici, aniž bychom cokoliv museli definovat. Podívejme se, jaké formáty se nabízejí.

ConTEXt je zajímavou alternativou, kde je řešena i sazba na řádkový rejstřík. Jeho principiálním nedostatkem je však to, že současná verze vyžaduje luaTEX. Pro sazbu hindštiny je tudíž tento formát nepoužitelný.

Plain TEX je výhodný zejména proto, že nezatěžuje počítač hromadou maker, která v dokumentu stejně nebudou využita. V době přípravy systému pro sazbu zmíněné knihy však neexistoval pdfcsplain v dnešní podobě, nebyl tedy k dispozici ortogonální systém volby fontů, nebyla podpora pro sazbu ve více jazycích. To vše si uživatel pro každý dokument musel naprogramovat *ab initio*.

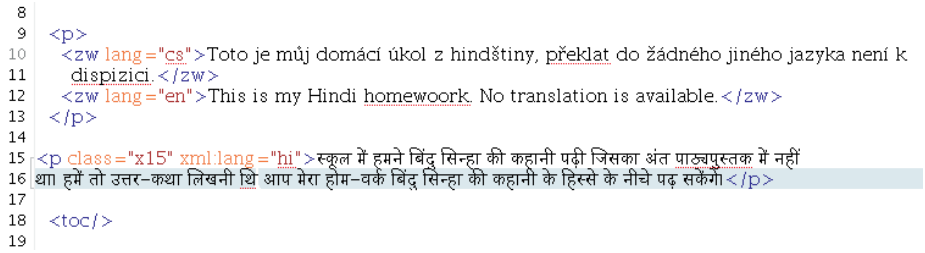
LATEX nabízí mnoho standardních maker a rozsáhlou kolekci balíčků. Podporuje především pohodlnou práci s fonty a tvorbu vícejazyčných dokumentů. Řadu maker sice nebudeme potřebovat, za což zaplatíme pomalejším zpracováním, ale na současných počítačích to není problém. XELATEX, tedy formát LATEX spojený se sázecím strojem XETEX, je tudíž přirozenou volbou.

Zastavme se ještě u formátu pdfcsplain [6]. Existují totiž tři stejnojmenné formáty, první pro osmibitový TEX, druhý pro luaTEX a třetí pro XETEX. Zatímco informace, že se má daný dokument zpracovat pdfLATEXem, luaLATEXem nebo XELATEXem, je zcela jednoznačná, informace, že zpracování dokumentu vyžaduje pdfcsplain, je nejednoznačná. Je nutno ještě specifikovat požadovaný sázecí stroj. Pokud bychom využili pdfcsplain se sázecím strojem XETEX a makrem OPmac [7], měli bychom pravděpodobně k dispozici vše potřebné. Tyto nástroje však byly oficiálně představeny až v době, kdy byla kniha již vytištěna, a neoficiální informace byla zveřejněna v okamžiku, kdy byla napsána nadpoloviční část knihy.

2.4. Volba způsobu zápisu

Běžná kniha (beletrie) obsahuje prostý text s malým počtem strukturních značek, vizuální značky při správném stylu zápisu nepoužíváme. V takovém textu nehrozí velké riziko chybovosti a pokud chybu uděláme, většinou její příčinu odhalíme snadno. Naproti tomu obrazová publikace je založena na komponování vizuálního vzhledu stránek. Jádrem zdrojového textu budou ryze vizuální značky určující umístění obrázků a makra pro vymezení textových oblastí. Tato makra budou mít mnoho parametrů, v nichž lze snadno udělat chybu. Podívejme se na příklad chybného kódu:

```
\clearpage
\null
\vfill
\thispagestyle{enpty}
\begin{picture}(0,0)
...
\end{picture}
```



Obrázek 2: Vícejazyčná kontrola pravopisu

Bez ořezových značek tento kód fungoval, ale při zapnutí ořezových značek v balíčku ZWPAGELAYOUT[1] došlo k nekonečnému cyklu a k havárii, která způsobila, že nebyl vygenerován žádný výstup. Na těchto několika řádcích se chyba najde poměrně snadno, ale hledání jednoho špatného písmenka v dlouhém dokumentu je komplikované. Potíž je v tom, že `\thispagestyle` volá definici stránkového stylu s použitím primitivu `\csname`, takže není nahlášeno nedefinované makro. Externí validátor se pro TEX píše velmi nesnadno. Uživatel si totiž může napsat vlastní makra, dokonce vlastní stylový soubor, který umístí do adresářového stromu `texmf-local`. Takový validátor pak neví, zda je někde definován stránkový styl *enpty*, nebo zda se jedná o překlep. Je obecně známo, že *only TEX can read TEX.*

Zápis v XML nabízí dvě výhody. První výhodou je validace standardními prostředky. Nejde jen o kontrolu, ale při použití validujícího editoru vidíme chybu okamžitě při psaní souboru. Ve skutečnosti nabízejí validující editory kontextovou nápovědu a značky lze vybírat z menu. Posun kurzorovými šipkami v menu je snadný a rychlý, není nutno odsouvat ruku z klávesnice na myš. Tato nápověda proto psaní nezpomaluje, ale zrychluje.

Druhá výhoda je ukázána na obrázku 2. Tato výhoda však není obecným rysem, ale je implementována v komerčním editoru `<oXygen/>` [16]. Tento editor totiž automaticky přepíná jazyk pro kontrolu pravopisu podle hodnot atributů `lang` a `xml:lang`. Je to demonstrováno na zdrojovém textu webové stránky s povídkou [9], kde bylo úmyslně uděláno několik chyb. V hindštině má správně být थी místo थि, slovo पाठ्यपुस्तक je správné, ale není ve slovníku.

Vzhledem k těmto vlastnostem byl vybrán způsob zápisu v XML. Protože pro sazbu chceme použít XHLATEX, provedeme transformaci pomocí XSLT verze 2.0 [3]. Obdobný postup již byl použit při sazbě jiné knihy [13].

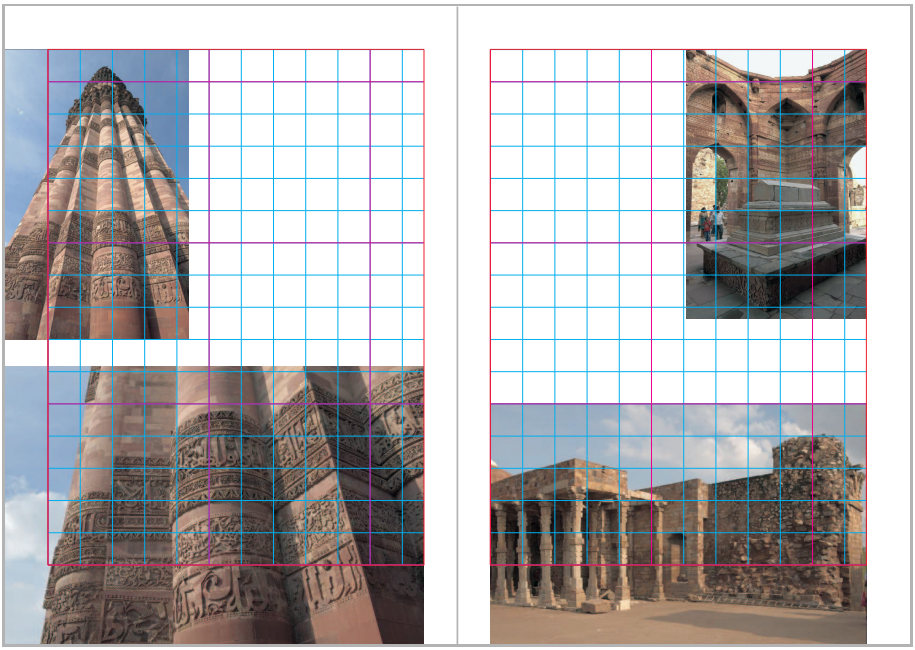
[1] Balíček byl již aktualizován a při detekci této chyby vypíše srozumitelné hlášení.

3. Pracovní postup

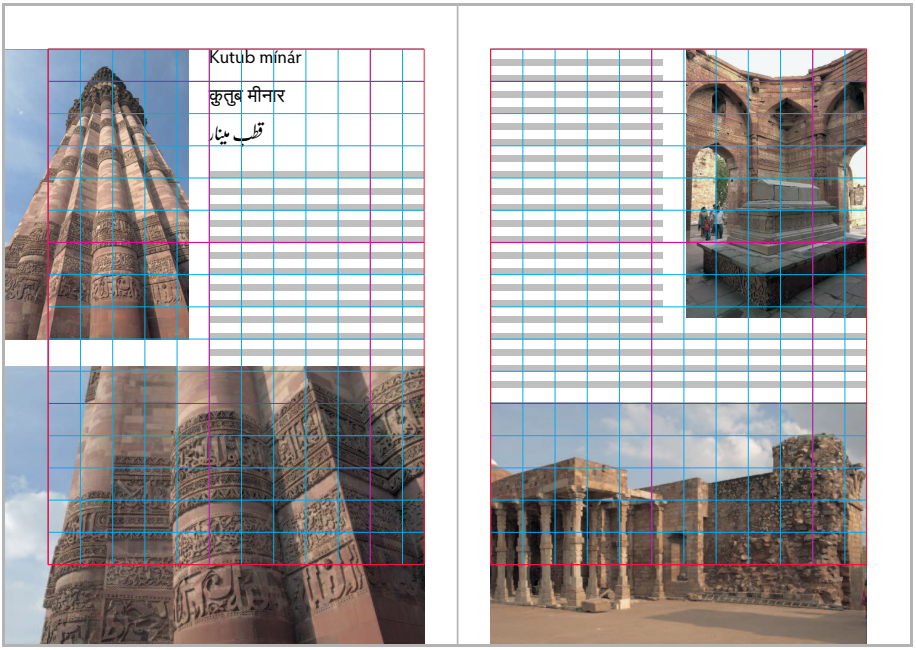
Z logiky popisu nástroje se zdá vysvětlování pracovního postupu na prvním místě nevhodným. Z hlediska vývoje však je obvyklé, že nejprve navrhneme požadovaný pracovní postup, poté vytvoříme a otestujeme základní kameny, které nakonec spojíme do funkčního celku. Ukázky prezentované v této kapitole vznikly právě během tvorby a ladění stavebních kamenů, kdy klademe důraz na funkčnost a ověření všech požadovaných vlastností, nikoliv na estetiku.

Již bylo uvedeno, že těžištěm knihy jsou fotografie a jejich rozvržení na stránky. V prvním kroku tedy chceme pouze umístit fotografie. V některých případech budeme fotografie umisťovat na spad. Přitom nestačí jen kontrola vzhledu, ale budeme muset i poměřovat rozměry. Pro tento účel si zobrazíme mřížku. Jednotlivé linky mřížky jsou od sebe vzdáleny 15 mm (čistý formát knihy je A4). Příklad dvojstrany s fotografiemi ukazuje obrázek 3.

V druhém kroku si vymezíme textové oblasti. Potřebujeme je vidět ještě dříve, než začneme psát text. Toto zobrazení se dá realizovat různými způsoby. Vzhledu textu nejlépe odpovídá způsob, kdy jsou řádky textu nahrazeny tlustými šedými linkami. Jak si ukážeme později, právě tento způsob se v TEXu implementuje



Obrázek 3: Rozvržení fotografií na dvojstraně

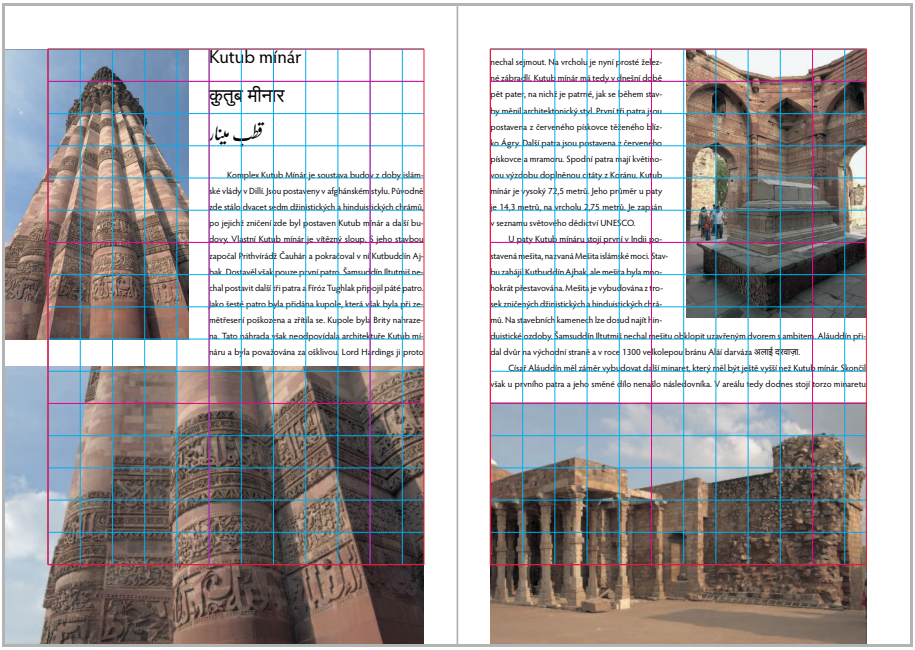


Obrázek 4: Zobrazení textových oblastí.

velmi snadno. Textové oblasti tedy vymezíme způsobem dle obrázku 4. Všimněte si nestejné šířky sloupců – na levé stránce 10 cm, na pravé stránce 8 cm. V této fázi již máme zobrazen nadpis kapitoly. Nemá totiž smysl vymezovat prázdnou oblast pro nadpis, protože ten je dán ještě dříve, než začneme vybírat fotografie pro kapitolu. Někdy dokonce vkomponujeme nadpis do fotografie a tehdy nutně musíme vidět náhled ještě před vymezením textových oblastí.

Třetím krokem je napsání vlastního textu. Může se stát, že výsledek nedopadne podle očekávání. Vzhled stránek s textem nemusí vypadat stejně dobře, jak vypadal náhled rozvržení textových oblastí. Ponecháme proto pomocnou mřížku, abychom rozměry textových oblastí mohli ještě změnit. Dvojstranu s textem a pomocnou mřížkou ukázuje obrázek 5. Všimněte si, že stránkový zlom skutečně nastal uprostřed odstavce, jehož sazba pokračovala na následující straně ve sloupci jiné šířky.

Mřížku si zobrazujeme barevně, ale přesto není čtení textu v pomocné mřížce pohodlné. Jakmile jsme s uspořádáním textu a fotografií spokojeni, zobrazení



Obrázek 5: Kontrola napsaného textu

mřížky potlačíme, abychom mohli zkontrolovat text. Způsob zobrazení přitom definujeme pro každou kapitolu samostatně. Můžeme se proto vrátit i k předchozí kapitole, když usoudíme, že by potřebovala změnu.

Práce na kapitole vyžaduje více kompilací a kniha má řadu kapitol. Obrázky v tiskovém rozlišení by psaní knihy velmi zpomalovaly. Používáme proto náhledy v malém rozlišení a v barevném prostoru RGB. Fotografie v plném rozlišení konvertované do barevného prostoru CMYK použijeme až ve finální verzi.

4. Implementace

V předchozí kapitole jsme stanovili pracovní postup. Nyní začneme hledat způsob, jak zdrojový text napsat a jak jej zpracovat. Pro určení formátu zápisu však musíme vědět, jaké informace bude sázecí stroj potřebovat. Při programování tedy budeme postupovat odzadu. Nejprve si připravíme TEXová makra. Pak budeme

postupovat zepředu, určíme si způsob zápisu v XML. Posledním krokem bude naprogramování transformace.

4.1. TEXové řešení

Idea řešení je vypůjčena z článku [10]. Ten však řeší mnoho případů, které v této knize nastat nemohou, a neřeší situace, které jsou pro naši konkrétní aplikaci typické. Proto byla stejná idea naprogramována od počátku jinak.

Obtékání obrázků je v podstatě záležitost nastavení vhodného tvaru odstavce. Není překvapením, že hotové balíčky pro LATEX se hodit nebudou. Naopak makro `\oblom` [5] je po malé úpravě použitelné.

Klíčovým prvkem bude primitiv `\vsplit`. Odlomené části budeme usazovat v LATEXovém prostředí `picture`, tedy uvnitř skupiny. Může se stát, že uvnitř jednoho prostředí `picture` budeme usazovat dvě odlomené části. Nemůžeme si tedy text nalámat na části předem, to by vyžadovalo komplikovanější makra. Naštěstí to není nutné, jak je vysvětleno v diskusním příspěvku [11].

Každá stránka je složena z obrázků a z textových oblastí. Všechny textové oblasti jsou buď obdélníkové, nebo je lze na několik obdélníků rozdělit. Sazbu tedy provedeme tak, že všechny textové oblasti v kapitole spojíme do jednoho boxu s nepravidelným tvarem. Box naplníme textem a pomocí primitivu `\vsplit` získáme jednotlivé části, které umístíme na správná místa na stránkách. Začneme přípravou registrů a pomocného makra:

```
\newbox\ZWbox           % pracovní box
\newbox\ZWsectionbox    % zásobník textů
\newcount\ZWcount
\newcount\shapenum      % tvar odstavce
\newcount\globpar
\newtoks\shapetoks
\newif\ifgrid           % mřížka
\def\ZWstrut{\vrule height \splittopskip width \z@}
```

Další makro slouží k definici tvaru složeného boxu pro všechny texty v kapitole. Za klíčovým slovem `\ZWXinitbox` budou následovat trojice *počet,odsazení,šířka;*, vše bez mezer, hodnoty odsazení a šířky se zadávají stejným způsobem jako všechny TEXové rozměry. Počet trojic není omezen, konec definice tvaru boxu je označen tokenem `\ZWinitbox`. Při zpracování každé trojice se počet řádků přičte do registru `\shapenum` a do `\shapetoks` se vloží odpovídající počet dvojic určujících odsazení a šířku řádku. Primitiv `\ifcat` při testu kategorie následujících dvou tokenů provádí úplnou expanzi. Tento typ testu si můžeme dovolit, protože TEXová matematika není v seznamu parametrů povolena. Pokud je tedy podmínka splněna, znamená to, že `#4` je prázdný. V takovém případě je zpracování parametrů ukončeno a do `\next` je vloženo `\doshape`. Pokud čtvrtý parametr není prázdný,

podmínka nebude splněna. Text se proto bude ignorovat a provedou se příkazy mezi primitivy `\else` a `\fi`. Zde je makro `\next` nadefinováno tak, aby vynutilo zpracování další trojice parametrů.

```
\def\ZWXinitbox{\shapenum=0 \shapetoks={}\ZWinitbox}
\def\ZWinitbox#1,#2,#3;#4\ZWinitbox{\globpar#1
  \advance\shapenum\globpar
  \loop
    \shapetoks\expandafter{\the\shapetoks #2 #3 }
    \advance \globpar -1
  \ifnum \globpar>0
  \repeat
  \ifcat $#4$\let\next\doshape\else
  \def\next{\ZWinitbox#4\ZWinitbox}\fi\next}
```

Makro `\doshape` je převzato beze změny z makra `\oblom` [5]:

```
\def\doshape{\globpar=0
    \def\par{\ifhmode\shapepar\fi}}
\def\shapepar{\prevgraf=\globpar \relax
    \parshape \shapenum \the\shapetoks \endgraf
    \globpar=\prevgraf
    \ifnum\prevgraf>\shapenum
          \let\par\endgraf \fi}
```

Sazba linek pro vizuální kontrolu textových oblastí je velmi jednoduchá. Použijeme hodnoty, které jsme si uschovali při definici tvaru složeného textového boxu. Makro `\ZWrules` tedy vygeneruje odpovídající počet maker `\ZWrule` obsahujících příslušné rozměry:

```
\def\ZWrules{\globpar\shapenum
    \noindent\ZWstrut
    \expandafter\ZWrule\the\shapetoks}
\def\ZWrule #1 #2 {%
    \textcolor{rules}{\vrule
       width #2 height 8pt}\penalty 0 \space
    \advance\globpar -1
    \ifnum\globpar>0\let\next\ZWrule\else
    \let\next\par\fi\next}
```

Všimněte si, že makro `\ZWrule` využívá pouze šířku, odsazení ignoruje. Odsazení je totiž definováno ve tvaru složeného boxu. Protože délka všech linek je přesně rovna šířce řádku v boxu, lámou se řádky v penaltě před mezerou. Penalta je tam zbytečná, bez ní by se řádky lámaly v mezerách.

Text kapitoly vložíme do prostředí `sectionbox`, jehož parametrem je definice tvarů textových oblastí, tedy posloupnost trojic *počet,odsazení,šířka;*. Do těla tohoto prostředí vložíme buď makro `\ZWrules`, nebo text:

```
\newenvironment{sectionbox}[1]{%
  \ZWXinitbox#1\ZWinitbox\global
  \setbox\ZWsectionbox=\vbox\bgroup
    \parindent 2em \lineskiplimit -12pt
    \relax}{\par\egroup}
```

Při sazbě stránky potřebujeme odlomit box dané výšky. Současně musíme nastavit požadovanou šířku, protože složený box má šířku určenou nejširším řádkem. Pokud bychom nenastavili původní šířku podle definice dané textové oblasti, mohl by box být umístěn na stránce nesprávně:

```
\def\ZWvbox{\ZWsplit\vbox}
\def\ZWvtop{\ZWsplit\vtop}
\def\ZWsplit#1#2#3{\setbox\ZWbox = #1{\vsplit
    \ZWsectionbox to #2mm}\wd\ZWbox=#3mm}
```

Popis vzhledu stránky se vkládá do prostředí `page`, které je definováno tímto předpisem:

```
\newenvironment{page}[1]{#1 \null
  \vfill \parindent\z@ \unitlength 1mm
  \begin{picture}(0,0)}%
  {\ifgrid\ZWgrid\fi\end{picture}}
```

Nepovinný parametr obsahuje kód, který se má provést na začátku stránky. Obvykle je to nastavení jiného typu pomocí makra `\thispagestyle`, ale může zde být vynucený přechod na sudou či lichou stránku využitím maker `\NewEvenPage` a `\NewOddPage` definovaných v balíčku ZWPAGELAYOUT. Stránka je sestavována pomocí prostředí `picture`, kdy makrem `\put` vkládáme postupně obrázky a texty odlomené ze zásobníku. Protože nadpis kapitoly může být vložen i do obrázku, musíme pomocnou mřížku vykreslit až na závěr. Příkaz pro tisk mřížky vyžaduje, aby rozměry a počet linek byly předem nastaveny na základě informací ze zdrojového souboru XML:

```
\def\ZWgrid{\color{cyan}
    \linethickness{.1pt}
    \multiput(0,0)(0,15){\horizlines}{\line(1,0){\ZWlinewidth}}
    \multiput(0,0)(15,0){\vertlines}{\line(0,1){\ZWTop}}
    \color{magenta}\linethickness{.2pt}
    \multiput(0,0)(0,75){\Horizlines}{\line(1,0){\ZWlinewidth}}
```

```
\multiput(0,0)(75,0){\Vertlines}{\line(0,1){\ZWTop}}
\color{red}\linethickness{.5pt}
\put(0,0){\framebox(\ZWlinewidth,\ZWTop)[lb]{}}
\color{black}}
```

4.2. Definice schématu XML

Jazyk XML byl vytvořen tak, aby jeho zpracování bylo možné i bez znalosti struktury, což v SGML z principu nelze. Nemuseli bychom se tedy definicí struktury vůbec zabývat, ale tím bychom se zbavili tří podstatných výhod. O dvou jsme se již zmínili, je to kontrola správnosti (validace) pomocí standardních nástrojů a kontextová nápověda ve validujícím editoru. O třetí výhodě se zmíníme v kapitole 4.3.

Svět XML nabízí dva jazyky pro popis struktury, které jsou funkčně velmi podobné, XML schema [22] a Relax NG [18]. XML schema umožňuje určení minimálního a maximálního počtu opakování elementu, zatímco Relax NG nabízí pouze možnosti `<optional>`, `<zeroOrMore>` a `<oneOrMore>`. Tuto nevýhodu lze však snadno obejít použitím Schematronu [19], k čemuž lze využít i NVDL [15]. Výhodou Relax NG je možnost specifikace nepovinné skupiny atributů. Přestože zde tato vlastnost nebude potřebná, zvolili jsme použití Relax NG. Použití Schematronu pro důkladnější kontrolu není nutné. Celé schéma zde nebudeme uvádět, ukážeme si pouze několik zajímavých částí.

Ve zdrojovém souboru budeme zadávat rozměry. Syntaktickou správnost chceme validovat, proto validátor musí znát způsob, jak se rozměry zadávají v TEXu. Rozměr můžeme zadat jako `.5cm`, `.5in` nebo `.5\linewidth`. Registr lze tedy použít jako jednotku, k níž přidáme desetinné číslo. Zatímco `\linewidth` bez číselné hodnoty je legitimní určení rozměru, jednotku `cm` samostatně použít nemůžeme. Abychom schéma nekomplikovali, budeme předpokládat, že výchozí číselná hodnota je vždy 1. O její doplnění se postará transformace. Definice rozměrů a jejich použití pro určení výšky a šířky pak vypadá takto:

```
<!-- Jednotky, které zná TeX -->
<define name="def.abs.unit">
  <choice>
    <value>mm</value>
    <value>cm</value>
    <value>in</value>
    <value>pt</value>
    <value>bp</value>
    <value>dd</value>
    <value>pc</value>
    <value>cc</value>
  </choice>
```

```
</define>

<!-- Relativní šířková jednotka,
     cropwidth = na spad od kraje do kraje -->
<define name="def.width.unit">
  <choice>
    <value>linewidth</value>
    <value>cropwidth</value>
  </choice>
</define>


<!-- Relativní výšková jednotka,
     cropheight = na spad od kraje do kraje -->
<define name="def.height.unit">
  <choice>
    <value>textheight</value>
    <value>cropheight</value>
  </choice>
</define>

<!-- Není-li uvedena číselná hodnota, předpokládá se 1 -->
<define name="attr.value">
  <optional>
    <attribute name="value">
      <data type="decimal"/>
    </attribute>
  </optional>
</define>

<!-- Šířka -->
<define name="def.width">
  <ref name="attr.value"/>
  <attribute name="unit">
    <choice>
      <ref name="def.abs.unit"/>
      <ref name="def.width.unit"/>
    </choice>
  </attribute>
</define>

<!-- Výška -->
```

```
<define name="def.height">
  <ref name="attr.value"/>
  <attribute name="unit">
    <choice>
      <ref name="def.abs.unit"/>
      <ref name="def.height.unit"/>
    </choice>
  </attribute>
</define>
```

Hlavním elementem knihy je `<book>`, který je definován takto:

```
<!-- @imgpath = kořenový adresář z obrázky
  @jpg = podadresář s obrázky ve formátu JPG v RGB
  @pdf = podadresář s obrázky ve formátu PDF v CMYK
  @mode = draft: zobrazuje oblasti bez textu, jpg/pdf: volba obrázků
  @grid = zapíná/vypíná pomocnou mřížku -->
<define name="elem.book">
  <element name="book">
    <attribute name="lang">
      <value>cs</value>
    </attribute>
    <attribute name="svnId">
      <data type="string">
        <param name="pattern">$Id.*$</param>
      </data>
    </attribute>
    <ref name="attr.grid"/>
    <attribute name="imgpath"/>
    <attribute name="jpg"/>
    <attribute name="pdf"/>
    <attribute name="mode">
      <choice>
        <value>draft</value>
        <value>jpg</value>
        <value>pdf</value>
      </choice>
    </attribute>
    <optional>
      <attribute name="title"/>
    </optional>
    <ref name="elem.sizes"/>
    <ref name="elem.layout"/>
```

```
    <ref name="elem.images"/>
    <ref name="elem.frontmatter"/>
    <zeroOrMore>
      <ref name="elem.part"/>
    </zeroOrMore>
    <ref name="elem.backmatter"/>
  </element>
</define>
```

V elementu `<sizes>` nadefinujeme skutečné velikosti písma pro LaTeXové přepínače `\normalsize`, `\small`, `\large` apod. Element `<layout>` obsahuje parametry vyžadované balíčkem ZWPAGELAYOUT. Z nich se též vypočtou hodnoty, které jsou nutné při použití makra `\ZWgrid`. Úvodní a závěrečnou část knihy nebudeme zapisovat v XML, ale pro ušetření práce raději přímo v LaTeXu. Elementy `<frontmatter>` a `<backmatter>` určí místa, kam budou odpovídající soubory načteny. Je zřejmé, že části knihy, které nebudou zapsány v XML, budou obsahovat i obrázky. Systém pro zpracování knihy však o nich musí vědět. Odkazy na tyto obrázky proto uvedeme v elementu `<images>`. O tomto elementu se podrobněji zmíníme v kapitole 4.3.

Vzhled stránky definujeme v elementu `<page>`. Následující ukázka vkládá na stránku nadpis kapitoly, dva obrázky a obdélníkovou textovou oblast. LaTeX vytiskne všechny objekty v tom pořadí, v jakém jsou uvedeny ve zdrojovém kódu XML. Pokud chceme tisknout text do obrázku, musíme obrázek uvést dříve než příslušný text.

```
<page thispagestyle="empty">
  <titlebox halign="right" textalign="left">
    <x>right</x> <y>Top</y>
    <height unit="cm" value="3"/>
    <width unit="cm" value="10"/>
  </titlebox>
  <img src="830/img101" pos="lt">
    <height value="13.5" unit="cm"/>
    <x>outer</x> <y>Top</y>
  </img>
  <img src="830/img097" pos="lb">
    <width unit="mm" value="200"/>
    <x>outer</x> <y>BOTTOM</y>
  </img>
  <text halign="right" valign="bottom">
    <x>right</x> <y>97.5</y>
    <height unit="cm" value="9"/>
    <width unit="cm" value="10"/>
```

```
  </text>
</page>
```

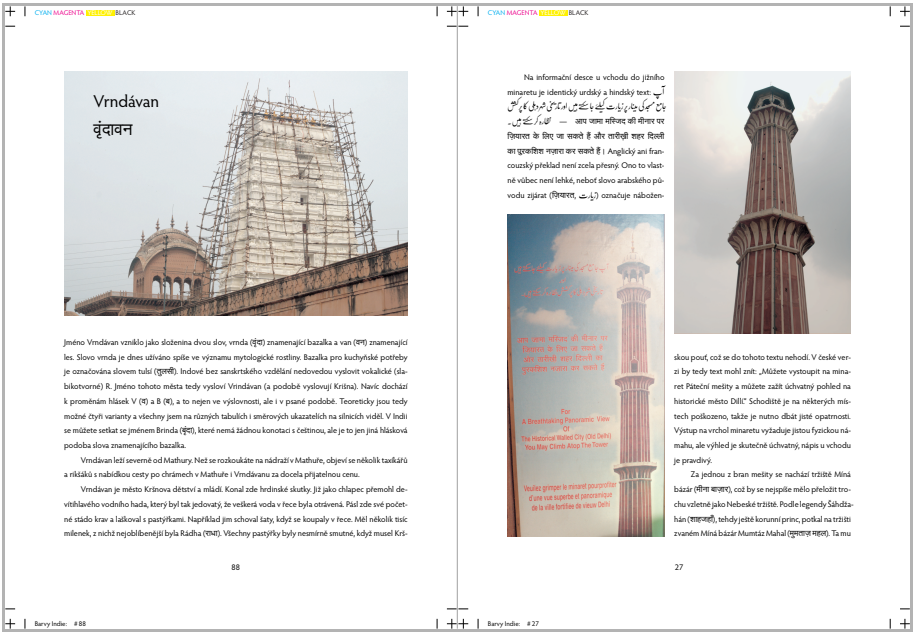
Pokud chceme definovat složitější textovou oblast, musíme v elementu `<text>` uvést několik elementů `<shape>`:

```
<text valign="bottom" halign="left">
  <x>left</x> <y>82.5</y>
  <shape>
    <indent value="0" unit="cm"/>
    <width unit="cm" value="8"/>
    <height unit="mm" value="127.5"/>
  </shape>
  <shape>
    <indent value="0" unit="cm"/>
    <width unit="linewidth"/>
    <height unit="mm" value="30"/>
  </shape>
</text>
```

Na obrázku 6 jsou ukázány dvě vybrané stránky knihy. Strana 88 obsahuje nadpis uvnitř obrázku. Na straně 27 jsou dva obrázky a dvě textové oblasti. Levý sloupec má šířku 8 cm, pravý 9 cm. Nejenže je zlom mezi textovými oblastmi uprostřed odstavce, ale dokonce je zde rozdělené slovo. Současně si můžete všimnout, jak se v odstavci, kde se na začátku píše zleva doprava, dělí na řádky víceřádkový urdský text, který se píše zprava doleva. XeLaTeX tuto úlohu zvládá správně.

Ukázka zdrojového textu v editoru je na obrázku 7. Je zřejmé, že i editor si poradí se správným zobrazením víceřádkového textu v urdštině. Není to však obecnou vlastností všech editorů. Můžete si ale všimnout, že písmena v urdštině nejsou správně spojena. Příčina problému spočívá v tom, že nemáme dostatečnou kontrolu nad tím, který font pro který blok Unicode editor použije. Zde si editor pro urdštinu vybral arabský font, který ale neobsahuje všechny znaky, jež urdština potřebuje. Chybějící znaky vzal z jiného fontu. Znaky, které pocházejí ze dvou různých fontů, však spojit nelze, proto urdské znaky zůstávají v izolovaných formách. Protože text bude pouze vysázen a nikdy nebude použit pro jiný účel, nezatěžujeme se korektním zápisem a v textu ponecháváme symboly, s nimiž si TeX poradí. Zde je to dvojice znaků ‘‘, v jiných částech se vyskytují vlnky pro vyznačení nezlomitelných mezer i `\,` pro vložení zúžené mezery.

Typ kapitoly může být `chapter` nebo `section`. Správnost vnoření není kontrolována. Typ kapitoly pouze určuje velikost fontu v nadpisu a způsob zápisu do obsahu.



Obrázek 6: Ukázka konečné podoby dvou stran s ořezovými značkami

4.3. Transformace XSLT

Zbývajícím krokem je transformace XSLT. Při popisu TEXové části jsme nadefinovali, jaký má být výsledek transformace, který musíme získat zpracováním zdrojového souboru XML. Zde se ukáže dříve zmíněná třetí výhoda použití validace. Schéma totiž jasně určuje, jaké případy ve zdrojovém souboru mohou nastat, vše ostatní je nepřípustné. Víme tedy, co je nutno v transformačních šablonách ošetřit.

Celý transformační styl zde uvádět nebudeme. Ukážeme si pouze funkci, která rozměr, zapsaný v XML, převede na TEXový zápis. V kapitole 4.2 jsme si řekli, že transformace se musí postarat o vložení hodnoty 1, pokud číselná hodnota uvedena není. To však není jediná úloha. Pokud v roli jednotky používáme TEXový registr, musíme vložit i zpětné lomítko. Rozhodování je jednoduché. Jména všech jednotek jsou dvoupísmenná, jména všech registrů mají více písmen.

```
<!-- Compose value and TeX unit, return 0mm on empty sequence -->
<xsl:function name="zw:TeXdim" as="xs:string">
  <xsl:param name="dim" as="element()?"/>
  <xsl:choose>
```

```
<part type="section">
  <title>
    <line lang="cs">
      <titletext>Džámá masdžid</titletext>
    </line>
    <line lang="hi">
      <titletext>जामा मस्जिद</titletext>
    </line>
    <line lang="ur">
      <titletext>جامع مسجد</titletext>
    </line>
  </title>

  <para indent="no">Džámá masdžid, doslova Páteční mešita, je známa i pod názvem Džámí masdžid.
    Myslím, že jsem to viděl napsáno i v dévanágarí jako <span lang="hi">जामी मस्जिद</span>.
    Kolísání slovních tvarů i pravopisu je v hindštině poměrně běžné. U slov arabského a perského
    původu dochází k odchylnostem v psaní spřežek. Slovo masdžid, mešita, je psáno <span
      lang="hi">मस्जिद</span> i <span lang="hi">मसजिद</span>, přičemž obě varianty jsou správné. V
    nadpisu této kapitoly je použita varianta, která je napsána přímo na jedné z informačních desek
    v mešitě.</para>

  <para>Džámá masdžid je největší mešita v Indii. Pojme až 25 tisíc lidí. Byla postavena v letech
    1644–1658. Má tři vstupní brány, čtyři menší věže, tři kopule a dva minarety vysoké čtyřicet
    metrů. Jižní minaret je přístupný veřejnosti.</para>

  <para>Na informační desce u vchodu do jižního minaretu je identický urdský a hindský text:
    <span lang="ur">آپ جامع مسجد کی
      مینار پر زیارت کی لئے جا سکتے ہیں اور تاریخی شہر دہلی کا بہترین نظارہ کر سکتے ہیں۔</span>\quad —\quad <span lang="hi">आप जामा मस्जिद की मीनार पर ज़ियारत के
      लिए जा सकते
      हैं और तारीख़ी शहर दिल्ली का पुरकशिश नज़ारा कर सकते हैं।</span> Anglický ani francouzský
    překlad není zcela přesný. Ono to vlastně vůbec není lehké, neboť slovo arabského původu zijárat (<span
      lang="hi">ज़ियारत</span>, <span lang="ur">زیارت</span>) označuje náboženskou pouť, což se do
    tohoto textu nehodí. V české verzi by tedy text mohl znít: „Můžete vystoupit na minaret
    Páteční mešity a můžete zažít úchvatný pohled na historické město Dillí.“ Schodiště je na
    některých místech poškozeno, takže je nutno dbát jisté opatrnosti. Výstup na vrchol minaretu
    vyžaduje jistou fyzickou námahu, ale výhled je skutečně úchvatný, nápis u vchodu je pravdivý.</para>

```

Obrázek 7: Ukázka zdrojového textu v editoru

```
    <xsl:when test="empty($dim)">
      <xsl:text>0mm</xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:variable name="value" as="xs:decimal"
        select="if ($dim/@value) then $dim/@value else 1"/>
      <xsl:variable name="unit-sep" as="xs:string"
        select="if (string-length($dim/@unit) &gt; 2)
                then '\' else ''"/>
      <xsl:value-of
        select="concat($value, $unit-sep, $dim/@unit)"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>
```

Konverze z XML do TEXu však není jedinou úlohou. Potřebujeme též zpracovat obrázky. Originály byly fotoaparátem uloženy ve formátu NEF. Programem VueScan [21] byly upraveny a uloženy ve formátu TIFF s bezztrátovou kompresí. Některé obrázky byly dodatečně oříznuty. Pro vkládání do sazby však potřebujeme buď náhledy v malém rozlišení uložené jako JPG, nebo finální podobu v barevném prostoru CMYK. Další transformační styl tedy ve zdrojovém souboru vyhledá názvy všech souborů s obrázky, které je nutno zpracovat. Tento styl hledá v elementu `<images>` názvy všech souborů s obrázky použitých v částech, jež jsou zapsány přímo v TEXu.

4.4. Automatizace zpracování

Transformaci a kompilaci XELATEXem lze spouštět ručně, ale konverze obrázků je náročnější. Jejich celkový počet ve formátu TIFF je více než 2 000, ale v knize jich nakonec bylo použito 296, což lze rychle zjistit výrazem `count(//img)`. Při ruční konverzi obrázků bychom jistě často chybovali. Pokud bychom chtěli konvertovat všechny fotografie bez výběru, bylo by první zpracování časově velmi náročné a potřebovali bychom velký diskový prostor. Proces je tedy nutno automatizovat.

Výhodným nástrojem pro práci s XML je Ant [20]. Přestože spouštění jiných programů je možné, v tomto případě se pro kompilaci XELATEXem hodit nebude. Ant nabízí úlohy `<depend>` a `<dependset>` pro nastavení libovolných závislostí, ale v našem případě budou výsledné soubory ve formátu PDF závislé na obrázcích, přičemž názvy souborů zjistíme až během transformace. Proto zkombinujeme dva nástroje. Ant spustí dvě nezávislé transformace. Při první vygeneruje soubor pro zpracování XELATEXem, druhá transformace vygeneruje `Makefile`, který bude použit pro konverzi obrázků a TEXovou kompilaci. Řídicí soubor pro Ant je velmi jednoduchý:

```
<?xml version="1.0" encoding="UTF-8"?>
<project basedir="." default="make">
  <property name="src" value="BharatKeRang.xml"/>

  <target name="auxcopy">
    <delete file="zwpath.sty.aux" failonerror="no"/>
    <exec executable="./auxcopy.sh"/>
  </target>

  <target name="xslt.make"
   description="Build the Makefile" depends="auxcopy">
    <xslt basedir="." style="Makefile.xsl"
          in="${src}" out="Makefile"/>
  </target>
```

```
  <target name="xslt.tex"
          description="Build the XeLaTeX source text"
          depends="auxcopy">
    <xslt basedir="." style="xml2tex.xsl"
          in="${src}" out="BharatKeRang.tex"/>
  </target>

  <target name="make" depends="xslt.make,xslt.tex"
          description="Build all">
    <exec executable="make">
      <arg value="dirs"/>
    </exec>
    <exec executable="make">
      <arg value="-j"/>
      <arg value="3"/>
    </exec>
    <delete file="zwpath.sty.aux" failonerror="no"/>
  </target>
</project>
```

Za zmínku stojí, že i *make* se v něm pouští dvakrát. Při prvním spuštění se generují potřebné adresáře pro obrázky. Protože při psaní knihy se můžeme rozhodnout, že obrázky, které jsme chtěli použít, nakonec nevložíme, při prvním použití programu *make* se tyto nadbytečné soubory smažou, aby na disku nezabíraly místo. Zvláštní úlohu plní skript `auxcopy.sh`:

```
#!/bin/bash
# $Id: auxcopy.sh 178 2012-09-30 21:48:37Z zw $

sty=zwpath.sty
aux=$sty.aux

if [ -f $sty ]
then
  if [ -f $aux ]
  then
    diff -q $aux $sty && mv -v $aux $sty || rm $aux
  else
    cp -pv $sty $aux
  fi
else
  echo '%'>$aux
fi
```

Při transformaci totiž vzniká mimo jiné soubor `zwpath.sty`, na němž závisí tvorba všech PDF souborů. Tento soubor se však zřídka mění. Při tvorbě obálky a předsádek by se zbytečně sestavovala celá kniha. Tento skript proto nejprve vytvoří kopii souboru `zwpath.sty`. Při spuštění cíle `dirs` tento skript zkontroluje, zda se nově vygenerovaná verze liší. Pokud ne, nahradí se zkopírovanou verzí, takže čas vytvoření zůstane zachován. Jako poslední akce programu Ant je uschovaný soubor smazán. Závěrečné smazání uschovaného souboru i příkaz `echo '%'>$aux` řeší případ, kdy soubor `zwpath.sty` ještě neexistuje, nebo předchozí sestavování skončilo chybou a soubory jsou poškozeny.

5. Závěr

Vytvořený systém demonstruje, jak lze spojení nástrojů využít pro efektivní práci, kdy jsou syntaktické chyby detekovány již ve fázi editace zdrojového souboru a kontextová nápověda usnadňuje orientaci v rozsáhlém repertoáru atributů. Soubory potřebné pro sazbu takové knihy mají méně než 600 řádků schématu Relax NG, méně než 700 řádků XSLT a méně než 100 řádků TEXových maker, zbytek je ve standardních balíčcích, které jsou obsaženy v distribucích MiKTEX i TEX Live. Přestože je rozsah schématu Relax NG a transformačního stylu XSLT zdánlivě objemný, jedná se opět o soubory XML s pevnou syntaxí, takže jejich tvorba s použitím validujícího editoru je snadná a rychlá. Všechny použité soubory včetně příkladu najdete na webové stránce knihy, pro jejíž sazbu byly použity [12].

TEXová část řešení našla později uplatnění i při sazbě dvojjazyčné sbírky poezie [8]. Texty byly opět uschovány v boxech a následně nalámány na stránky tak, aby vytvořily zrcadlový překlad.

Reference

[1] Klaas Bals, Tony Graham: Požadavky na XSL-FO verze 2.0. *Zpravodaj Československého sdružení uživatelů TEXu* **20**(1–2), 79–120 (2010). DOI 10.5300/2010-1-2/79.

[2] Tomáš Hála: Osobní sdělení. TEXperience, Morávka 2012.

[3] Michael Kay (ed.): *XSL Transformations (XSLT) Version 2.0.* W3C Recommendation, 23 January 2007. [cit. 2014-03-22] `http://www.w3.org/TR/xslt20/`

[4] Jiří Kosek: PassiveTEX. *Zpravodaj Československého sdružení uživatelů TEXu* **13**(1), 26–38 (2003). DOI 10.5300/2003-1/26.

[5] Makro `\oblom`. *In:* Petr Olšák: TEXbook naruby, kap. 6, str. 236–237. Konvoj 1997. ISBN 80-85615-64-9.

[6] Petr Olšák: Nový csplain. *Zpravodaj Československého sdružení uživatelů TEXu* **12**(1), 42–58 (2012). DOI 10.5300/2012-1/42.
[7] Petr Olšák: OPmac – makra rozšiřující možnosti plainTEXu. *Zpravodaj Československého sdružení uživatelů TEXu* **12**(1), 20–41 (2012). DOI 10.5300/2012-1/20.
[8] Aruna Rai, Zdeněk Wagner: अरुणाकाश का रास्ता / Cesta do červánků. Nakladatelství Zdeněk Vavřínek, Praha 2013. ISBN 978-80-905324-2-7.
[9] Bindu Sinha, Zdeněk Wagner: सुबह , दोपहर और शाम. [cit. 2014-03-22] `http://icebearsoft.euweb.cz/rachna/subah.dopahar.sham.php`
[10] Jan Šustek: Sazba odstavců do textových oblastí. *Zpravodaj Československého sdružení uživatelů TEXu* **19**(3), 124–137 (2009). DOI 10.5300/2009-3/124.
[11] Jan Šustek: Je `\vsplit` opravdu globální? `http://lists.felk.cvut.cz/pipermail/cstex/2011-October/024285.html`
[12] Zdeněk Wagner: *Barvy Indie*. Ice Bear Soft, Praha 2012.
[13] Zdeněk Wagner: Využití XML a LATEXu při sazbě odborných knih, *Zpravodaj Československého sdružení uživatelů TEXu* **12**(3–4), 188–211 (2002). DOI 10.5300/2002-3-4/188.
[14] Helena J. Wagnerová: *Úniky z temna*. Martin, Brandýs nad Labem 2006. ISBN 80-85955-32-6.
[15] ISO/IEC 19757-4 NVDL (Namespace-based Validation Dispatching Language). [cit. 2014-04-13] `http://www.nvdl.org/`
[16] `<oXygen/>` xml editor. [cit. 2014-03-22] `http://www.oxygenxml.com/`
[17] Petr Olšák. [cit. 2014-03-17] `http://petr.olsak.net`
[18] Relax NG home page. [cit. 2014-04-13] `http://relaxng.org/`
[19] Schematron. [cit. 2014-04-13] `http://www.schematron.com/`
[20] The Apache Ant Project. [cit. 2014-04-13] `http://ant.apache.org/`
[21] VueScan Scanning Software. `http://www.hamrick.com/`
[22] XML schema current status. [cit. 2014-04-13] `http://www.w3.org/standards/techs/xmlschema`

Summary: Typesetting an Illustrated Publication with a Floating Text

The algorithms of TEX are intended for text processing, they are not designed to work with external images. This work is left to the output drivers. The drivers follow evolution of polygraphy. Thus especially by using the GRAPHICX package, images in common formats can easily be included in a way independent of the output driver used. The LATEX format is primarily aimed at typesetting texts containing a few tables or figures. The mechanism of floating objects ensures that the tables and figures automatically find a convenient place within the text.

Another situation arises if the main part of the publication consists of pictures the position of which is bound to a strict place of a page and the text should float around them. Moreover, it is required that the text floats to the next page in the middle of a paragraph and then be typeset to a column of different width. In such a case the existing approach cannot be used and a set of appropriate macros must be developed. The article shows how XML and XeLaTeX with a special set of macros was used to typeset an illustrated publication.

Key words

floating text, XeLaTeX, XML, XSLT, validation.

Zdeněk Wagner
Ice Bear Soft
http://icebearsoft.euweb.cz