XML A INFORMAČNÍ SYSTÉMY[1]

Tomáš Pitner

Masarykova univerzita v Brně, Fakulta informatiky, Botanická 68a, 602 00 Brno
e-mail: tomp@fi.muni.cz

Abstrakt

V úvodu bude podán stručný přehled hlavních s XML souvisejících standardů: jmenné prostory, prostředky specifikace schématu XML jazyků, jazyky transformací a stylů a budou též diskutovány záležitosti bezpečnosti XML infrastruktury. Hlavní část příspěvku se bude věnovat problémům výstavby komponentních systémů na bázi XML a zmíněny klíčové problémy, které stojí před vývojáři i teoretiky v této oblasti a bude též zaměřena na současné možnosti využití XML v aplikacích, zejména pak internetových.

1. Přehled XML standardů a technologií

1.1 XML 1.0

XML *(eXtensible Markup Language)* značkovací jazyky jsou jazyky definované podle (zatím neinovované) specifikace XML 1.0 přijaté konsorciem W3C jako *W3C Recommendation* v lednu 1998, [XML98].
Ideově jde u XML o pokračování vcelku úspěšného – a jako ISO norma zavedeného – standardu značkovacích jazyků SGML *(Standard Generalized Markup Language)* pocházejícího z osmdesátých let.
Obecným cílem značkovacích jazyků je poskytnout nástroj k vytváření semistrukturovaných dat ležících na pomezí klasických *strukturovaných* („databázových“) dat, modelovaných podle hierarchických, síťových, relačních či objektových modelů, a *dokumentů*, tj. dat s variabilní strukturou.

1.2 Jmenné prostory – Namespaces

Prostory jmen (Namespace) jsou věcí poměrně běžnou v řadě programovacích jazyků (např. C++, evt. Java), kde vyjadřují okruh dostupnosti identifikátorů a tím logicky seskupují jednotlivé definice/deklarace. Podobná idea samozřejmě provází i značkovací jazyky – smyslem je dát logický význam *(qualification)* jednotlivým skupinám elementů a atributů – prostorům jmen - a poskytnout možnost kombinovat elementy a atributy z různých prostorů jmen (někdy též označovaném jako *slovník značek – markup vocabulary*) v jednom dokumentu či dokonce v jednom elementu, jedná-li se o atributy z více jmenných prostorů.
S pomocí prostorů jmen je možné uvažovat o aplikacích, jež z daných XML dat zpracovávají jen elementy, resp. atributy z určitých prostorů jmen, zatímco ostatní budou buďto ignorovat, zanechávat bez úprav nebo považovat za chybná.

1.3 Modelování XML dat a jazyky schémat

Ověření syntaktické správnosti XML dokumentu sestává ze dvou stupňů – první, zcela základní podmínkou je tzv. *správné utvoření (well-formedness)*, které je nezávislé na

[1] Tento příspěvek vznikl s podporou grantu GAČR 201/00/D110.

konkrétním jazyce založeném na XML. Druhou podmínkou je *validita (validity)*, která se již váže ke konkrétnímu jazyku (tj. *typu dokumentu*). *Validní* dokument splňuje, zhruba řečeno, podmínky specifikované příslušným *schématem* dokumentu, například, které elementy se mohou do daného elementu vnořit (případně i různě podle kontextu) a podobné podmínky pro atributy. Standardním prostředkem definování schémat je od XML 1.0 (a převzatým z SGML) tzv. *Document Type Definition (DTD)*.

DTD je se samotným XML 1.0 natolik pevně svázán, že i odkaz v *deklaraci typu dokumentu* <!DOCTYPE *root* PUBLIC *public_id* SYSTEM *system_id*> směřuje na DTD, čili pojem *validity* splývá v běžném kontextu s *validitou podle DTD daného v deklaraci typu dokumentu*. Nebudeme zde detailně studovat syntaxi ani sémantiku jednotlivých deklaračních konstrukcí, jejich popis lze najít v mnoha zdrojích, vč. samotného XML standardu [XML1.0], dále např. v [PokRi00]. Spíše se podívejme na omezení, které nám při datovém modelování DTD přináší:

- Nejsou k dispozici žádné primitivní datové typy kromě typu #PCDATA (≈ String) pro textové obsahy elementů a hodnoty atributů.
- Celkem je k dispozici 10 vestavěných datových typů, mezi nimi: PCDATA (≈ String), ID (jednoznačný v rámci dokumentu, může sloužit jako primární klíč), IDREF (odkaz na ID, je však netypovaný – nelze říci, na jako element se ukazuje), IDREFS (dtto, ale množina odkazů), NMTOKENS (seznam jmen), ENTITY, ENTITIES (odkazy na jednu/více neanalyzovaných entit).
- Není žádná možnost vytvářet vlastní typy a neexistují žádné kolekce (set, bag, sequence...).
- U hodnot atributů mohou být přípustné hodnoty stanovené výčtem (vč. default hodnoty).
- Nelze definovat integritní omezení – až na odkazy IDREF, IDREFS, které se odkazují na primární klíč/e nějakého/ých elementu/ů.

1.4 Co očekáváme od nového modelovacího jazyka?

Diskuse kolem nástupce nedostatečného DTD se rozběhly do několika směrů, především se lišících podle základního náhledu na:

- podporu užívaných *primitivních datových typů* (řetězce, celá a floating-point čísla různých rozsahů a přesností, booleovské hodnoty); možnosti *konstrukce vlastních typů* a mechanismus *odvozování* (dědičnost); možnosti specifikace *modelu obsahu elementů* (tzv. *content model*);
- možnost specifikace *integritních omezení* a možnost podpory *dotazovacích jazyků;*
- podporu *jmenných prostorů;*
- základní techniku ověřování (validace) souhlasu datové instance se schématem; *složitost validace* – algoritmickou a technickou;
- konceptuální *zpětnou kompatibilitu s DTD* a jeho „oficiálním“ nástupcem z dílny W3C: *XML Schema.*

Nelze říci, že by mezi pokročilými jazyky popisu schémat bylo zcela jasno, pokud jde o nástupnictví po DTD. „Oficiální“ standard *XML Schema*, publikovaný W3C definuje sice jakýsi „etalon“, s nímž se ostatní srovnávají a je také nejpodporovanější pokud jde o validační nástroje (zejména parsery), ale není bez nevýhod.
V současnosti jsou tedy (alespoň v teorii) používanými jazyky schemat (podrobné technické srovnání jazyků schémat je v [BoLe01]).:

* *XML Schema* (autor W3C, verze 1.0) pokrývá všechny možnosti DTD, přidává zejména řadu primitivních dat. typů, možnost rozšiřování děděním vlastností, podporuje některá integritní omezení;
* *Schematron* (autor Rick Jelliffe, nyní verze 1.5) postavený na validaci pomocí přijímání/odmítání XPath vzorů (viz [Je01]), což umožňuje validace v *kontextu*, dobrý popis modelu obsahu a podporu jmenných prostorů, nemá vestavěné prim. typy, lze však rozšířit přidáním uživatelských funkcí;
* *Relax* (pocházející z Japonska – autor Makoto Murata a standardizovaný ISO/IEC jako DIS 22250-1), podporuje některá integritní omezení (méně než XMLS), má bohaté primitivní datové typy;
* *SOX* (autor Commerce One, verze 2.0) vychází z DTD, které objektově rozšiřuje;
* *DSD* (autor AT&T, BRICS, verze 1.0) konceptuálně podobný Schematronu, klade důraz na validace s pomocí omezení specifikovaných vzory.

1.5 Dotazovací jazyky

Data kódovaná v XML jsou *semistrukturovaná*, což může v praxi znamenat, že vzniknout přímo *transformací z relačních tabulek* či *serializací* nějaké *objektové struktury*, nebo naopak vzniknout *označkováním* původně *nestrukturovaného dokumentu*. Stále častěji se budeme setkávat i s XML daty vzniklými smíšením těchto postupů – např. u Microsoftem prosazovaných *XML Data Islands*, viz [MS01]. U dotazování se tedy kombinují přístupy známé z obou oblastí:

- přístup jako k přísně *strukturovaným* (relačním či objektovým) datům – à la SQL, OQL;
- *navigace ve stromové struktuře* XML dokumentu – jako v adresářové struktuře systému souborů;
- *boolské vyhledávání* jako v klasických databázích plných textů;
- vyhledávání na základě *ontologií* – i s využitím interních (in-line) či externích (out-of-line) metadat, obvykle modelovaných v RDF;
- *navigace v síťovém prostředí* a přístup k heterogenním zdrojům, zdrojům s předem neznámou nebo nedefinovanou strukturou;

Výsledkem dotazování je buďto *fragment* (množina uzlů, *nodeset*) XML dokumentu, nebo jiný konstrukt (např. výsledek agregační funkce) vzniklý aplikací „konstrukční“ části dotazu. Možnost efektivního dotazovaní nad XML dokumenty je podmíněna existencí takového silně typovaného jazyka schémat, podporující také specifikaci integritních omezení a umožňující podobné optimalizace jako u R-SŘBD a SQL. Takovými pokusy je např. velmi aktuální *UCM – Unified Constraint Modeling*, [UCM01] nebo poněkud akademičtější přístup *XDuce,* viz [XD00].

Zastřešující aktivitou nad dotazovacími jazyky, formulující obecné požadavky na dotazování, je pracovní skupina při W3C *XML Query*, http://www.w3.org/XML/Query, definující především *požadavky* na dotazovací jazyk a tím nastavující referenční podmínky pro srovnávání konkurujících si dotazovacích jazyků. Konkrétním standardem se již stala *XML Algebra.*

Mezi význačné představitele XML dotazovacích jazyků počítáme zejména:

- *XML-QL* (Deutsch, Florescu, Levy, Fernandez) – byl jedním z prvních, svou obdobnou vyjadřovací silou jako SQL slouží jako etalon pro ostatní, nebyl však konstruován s podporou silnějších jazyků schémat (tedy silného typování, integritních omezení, atd.);

- *XQL* (Microsoft) – založen na výrazech XPath, ale následné projekce výsledků mají jednodušší „výrazivo“ než XSLT;
- *Lorel* (Standford University) – původně pro dotazování nad semistrukturovanými daty ještě před XML;
- *XSLT* (W3C) – jako dotazovací jazyk se používá XSLT transformační jazyk, pro extrakci (SELECT) se uplatňují XPath výrazy v různých konstrukcích (apply-template select, for-each select, if test, …). Výhodou je kompatibilita se *Schematronem*, jež specifikuje schémata pomocí výrazů XPath, a dostupnost XSLT procesorů. Nevýhodou je přílišná bohatost konstrukcí, znesnadňující algoritmickou optimalizaci dotazů;
- *Quilt (IBM, Software AG, INRIA)* je nový dotazovací jazyk navržený v smíšeném akademicko-komerčním týmu, vychází z osvědčených XPath (navigace v stromě dokumentu) a XML-QL (vázané proměnné).

1.6 Styly – CSS, CSS2, XSL, XSLT

Poměrně propracovanou (a složitou) možností použít pro prezentaci označkovaného dokumentu tzv. *styly* disponoval i jazyk SGML, zde se jednalo o styly *DSSSL*. Úkolem stylů bylo přidávat elementům v označkovaném dokumentu tzv. *formátovací objekty* a jejich vizuální podobu.

Pro využití v prohlížečích byly DSSSL styly shledány jako příliš komplikované, a proto byl pro vyvinut podstatně jednodušší standard, pro účely webu však tehdy vcelku dostatečný – tzv. *kaskádové styly (Cascading StyleSheets – CSS)*. Tyto styly umožňují mj. formátování elementů v *kontextu*, společné formátování množiny elementů, s kaskádovými styly rovněž do HTML přibývá možnost přiřazovat elementy do *tříd (classes)* a těmto třídám pomocí CSS přiřazovat společné formátovací vlastnosti. Touto technikou se nahrazuje nemožnost *uživatelsky rozšiřovat vlastní formátovací jazyk*.

Toto formátování už je (zvláště v aktuální verzi -- *CSS Level 2*) závislé na výstupním médiu (obrazovka, zvuk, tisk, znakové konzola), podporuje přesnější popis rozmístění formátovacích objektů na stránkách výstupního média a jejich přesnou vizuální podobu (velikosti a řezy písem, jejich modifikace, …). HTML dokumenty je dnes možné (s pomocí zvláštních „prohlížečů“) prezentovat i ve *zvukové podobě.*

Pro obecné transformace XML dokumentů CSS styly samozřejmě nedostačovaly. Na scénu proto přišel *eXtensible Style Language (XSLT)* pro popis přiřazení formátovacích objektů elementům a *eXtensible Style Language Transformation (XSLT)* pro specifikaci velmi obecných transformaci XML dokumentů, viz [XSLT00].

1.7 Bezpečnost

Bezpečnost XML dat při komunikaci a ukládání je řešena v současnosti na poměrně rudimentární úrovni. Delší dobu existuje standard

* *XML Signature* (viz [XSig00], vyvíjí W3C + IETF), který specifikuje standardní techniky (velmi obecné) pro elektronické podpisování, kontrolu integrity a autentizaci komunikujících stran u výměny XML dokumentů.

Vývoj možností el. podpisu je úzce vázán na pokroky okolo specifikace standardizovaného *kanonického tvaru* XML dokumentů (souvisí s XML datovým modelem) a na tom, jaký formát schémat (XML Schema?) a metadatový rámec (RDF/RDFS?) se prosadí – neboť na

nich bude např. záviset, kam podpisy vzhledem k dokumentu umisťovat. V současnosti je horkou novinkou (konečně!) založení W3C pracovní skupiny pro tvorbu standardu

* *XML Encryption* pro standardizované šifrování XML dokumentů.

2. XML jako základ komponentně orientovaných technologií

2.1 Architektury moderních aplikací

Moderní vícevrstvé (multi-tier) architektury jsou dnes již standardem pro budování rozsáhlejších a přitom interoperabilních aplikací. Podívejme se na základní vrstvy těchto aplikací a na potenciální či už aktuální penetraci XML do těchto vrstev.

* vrstva *databázová* (database layer)
* vrstva *aplikační logiky* (business logic layer)
* *prezentační* vrstva (presentation layer)
* a samozřejmě příslušný *middleware*

2.2 XML a databázová vrstva aplikací

XML je vhodným formátem pro přenositelná data. Pro jeho použití v databázích je však potřeba zvážit, co vlastně požadujeme:

* Jde o integraci stávajících datových zdrojů a zdrojů mimo naše databáze?
* Jde o stávající webové informační zdroje?
* Jde o data dokumentové povahy?
* Máme k dispozici (stávající nebo nový) SŘDB podporující práci s XML?

Dalším limitujícím faktorem praktického využití XML v datové vrstvě je existenci dostupných, dostatečně výkonných, robustních databázových systémů pro efektivní ukládání získávání a manipulaci s XML daty.

2.2.1 "Přizpůsobené" a nativní XML databáze

Většina „velkých hráčů" v oblasti databází přizpůsobuje své produkty XML realitě, avšak ne vždy stejně zásadně. Některé SŘBD to řeší primitivně nestrukturovaným ukládáním XML elementů jako BLOB, někde se používá rozklad do relačních tabulek (často se ztrátou původního uspořádání elementů) nebo ukládáním do zcela speciálních objektových struktur, kde je třeba samozřejmě kompletně řešit efektivní ukládání, rušení, indexování, jsou problémy s referenční integritou apod.

Mezi nativní XML databáze patří některé projekty open-source, např. *dbXML* (autorem skupina *dbXML Group*, hostováno na Sourceforge.net) a *XDB* (autorem skupina Zvon.org), XML podporuje dále např. *IBM DB2 XML Extender, Oracle 8i, 9i, MS SQL Server 2000, Software AG Tamino, Lore* a další.

2.3 XML a aplikační logika

Zde je zatím průnik XML z pochopitelných důvodů nejpozvolnější – XML modeluje data *deklarativním* způsobem, aplikační logika je konstruovaná většinou *procedurálně*. Navíc je aplikační logika ve stávajících aplikacích poměrně stabilní součástí, takže není důvod něco narychlo měnit. Přesto je i zde průnik XML patrný, uveďme několik typických příkladů:

* *Metadata aplikací* – např. konfigurační a pomocné soubory jsou kódovány v XML. Tím se (i dynamické) změny konfigurací stávají i „strojově“ možné. Typickým příkladem jsou *J2EE aplikace* konfigurované výhradně XML soubory.
* *Metadata aplikačních prostředí (serverů) a middleware* – nejen konfigurace aplikací, ale i jejich provozního prostředí se děje přes XML soubory – např. u *J2EE aplikační serverů.* Např. přiřazení uživatelských oprávnění (credentials), konfigurace virtuálních WWW serverů, samotného EJB aplikačního jádra serveru.

Použití XML pro konfiguraci aplikací a serverů je ruku v ruce s jeho standardizací, zejména pokud jde o kompaktní aplikační platformy, jakými je např. *Java 2 Enterprise Edition* (J2EE), používající komponenty *Enterprise Java Beans* (EJB) a také *Microsoft .NET* platforma.
V budoucnu je možné očekávat takový pokrok v této oblasti, že lze počítat s aplikacemi dynamicky instalovatelnými a konfigurovatelnými – a to i za běhu (tzv. *hot-deployment*). To se může dít zcela automatizovaně, jen podle nastavených pravidel (bezpečnost, zajištění synchronizace přístupů, řízení zátěže, zajištění konzistence dat, apod.). Častěji než monolitické systémy budou vyžadovány přísně komponentně budované aplikace tvořící tzv. *sítě autonomních komponent* (Network of Autonomous Components - NAC), viz [KrŽe00].

2.3.1 Nové příležitosti pro ASP

Zde leží obrovský tržní potenciál pro další uplatnění:
* *Application Services Provider* (ASP) a *Application Hosting* – hostování aplikací bude snadnější než kdy předtím a může se dít víceméně automatizovaně;
* při B2B elektronickém obchodu bude možné, aby jeden z obchodních partnerů dynamicky „zapojil“ vlastní aplikaci do aplikačního serveru partnera a vzdáleně ji konfiguroval.

Konfigurace a chování serverů a aplikační logiky je tedy stále více řízeno deklarativním popisem konfigurace formulovaným v XML souboru.
Vhodnou platformou pro konstrukci aplikační logiky pracující s XML se jeví Java, neboť vyjádření „Java je přenositelný kód, XML jsou přenositelná data“ se již stalo známou a pravdivou frází. Většina nových či inovovaných XML standardů tedy najde díky vývojové pružnosti Javy svou implementaci právě v tomto jazyku.

2.3.2 Soumrak klasické aplikační logiky?

Naproti tomu se však ukazuje, že i aplikační logika sama, tj. objekty – dělníci systému, mohu být z některých domén pozvolna vytěsněni speciálními komponentami postavenými čistě na XML standardech – např. mnohé dotazovací a transformační operace nad daty lze v budoucnu reimplementovat na platformě dotazovacího/transformačního jazyka XSLT, viz [XSLT00], přičemž potřebné stavové informace ponesou fragmenty XML dat.

2.3.3 Nástup metaprogramování

Velký význam bude mít též komponentně orientované aplikace postavené na metaprogramování – „on-fly“ vytváření kódu podle momentálních potřeb (např. podle schématu XML dat na vstupu se vytvoří příslušná obslužná komponenta, která se následně aktivuje technikami „hot-deployment“). Zde se opět mimořádně hodí možnosti známé a používané v Javě (dynamické vytváření, překlad a zavádění tříd).

2.4 XML a prezentační vrstva

Použití XML v této vrstvě bylo jedno z prvních; zejména proto, že popularita webových informačních systémů přinesla nutnost konstrukce prezentační vrstvy na bázi HTML jazyka, jehož je XML de-facto pokračovatelem. Základní kritéria pro použití XML v této vrstvě zní:

* Budujeme tuto vrstvu „z ničeho“ nebo ze stávající prezentační vrstvy? Je stávající prezentační vrstva webová?
* Kolik podvrstev prezentační vrstva má nyní a má mít?
* Jde o možnosti multimodálního přístupu k týmž datům přes WWW (HTTP, HTML), proprietárního klienta, WAP?
* Jde i o přístup z jiné aplikace, nejen od klienta obsluhovaného přímo člověkem?

Mnoho aplikací (většina) je dnes budováno na internetových/intranetových standardech, kam v tomto kontextu patří zejména vše související s WWW. Obvyklými technologiemi budování prezentační vrstvy (mnohdy bohužel smíšené s aplikační logikou) jsou u takových aplikací technologie *dynamického webu*: realizované často jako vícevrstvé – s dělením na

* *stranu serveru* (SSI, proprietární řešení typu ISAPI DLL, skriptovací jazyky, Java Servlety a technologie „aktivních stránek“ – ASP, PHP a Java Server Pages) a
* *stranu klienta* (prohlížeče) – kde vládne skriptování v JavaScriptu, proprietárním VB Scriptu, dynamické HTML (vč. kaskádových stylů) nebo použití plnohodnotného aplikačního prostředí JVM – pro Java Applety.

2.4.1 Architektura uživatelských rozhraní s použitím XML

Ke kvalitnějšímu návrhu a implementaci této vrstvy je však vhodné postupovat systematičtěji, začít od *konceptuálního modelu uživatelského rozhraní* vycházejícího z identifikace jednotlivých *případů užití (Use Cases)* a *navigačního konceptuálního modelu (NCM).*

* *Konceptuální model*, specifikovaný v XML a nezávislý na *cílovém* (klientském) prostředí (prohlížeč HTML, mobilní zařízení, zvukový výstup, cizí aplikace…) i na *zdrojovém* prostředí (na serveru). Obsahuje popis abstraktních vazeb na aplikační logiku. Tento model přebývá na straně serveru a bude se podle potřeby buďto staticky (jednorázově, např. při instalaci aplikace) nebo dynamicky transformovat do následujících modelů. Transformační technikou může být XSLT.
* *Logický model* – opět popsán v XML, ale už *závislý na konkrétním serverovém prostředí*. Jeho součástí bude např. kolekce JSP stránek s popisem přechodů mezi nimi a s konkrétní specifikací vazeb na aplikační logiku (např. na komponenty EJB). Logický model bude stále nezávislý na cílovém (klientském) prostředí a bude přebývat na straně serveru. Mezi prakticky využitelné technologie zde patří: XUL (Xml User interface Language), XSLT styly, JSP spolu s uživatelskými knihovnami značek (JSP Taglibs).
* *Fyzický model* – bude záviset i na konkrétním klientském prostředí – bude odlišný pro různé prohlížeče a výstupní média. Bude získán nejen XSLT transformací, ale i aplikací CSS stylů. Tyto transformace budou částečně nebo úplně realizovány až na straně klienta. Mezi prakticky využitelné technologie zde patří: (X)HTML, CSS1/2 styly, skriptování na straně klienta – zejména JavaScript a Swing rozhraní u appletů. Zatím trochu nedoceněnou technologií je zde model COM+.

Výsledkem návrhu bude většinou vícevrstvé uživatelské rozhraní, skládající se ze specifikace konceptuálního modelu, řady parametrizovaných transformačních pravidel, závisejících na konkrétním serveru, jeho konfiguraci a klientech. Podstatná je zde volná, ale dobře

specifikovaná vazba na aplikační logiku, které nebrání pohodlné úpravě či pozdější výměně prezentační vrstvy.

3. Shrnutí, perspektivy

XML technologie pronikají do všech vrstev moderních (webových) informačních systémů. Od klasických webových aplikací, jejichž předností byla nezávislost na klientském prostředí a dobrá dostupnost, se postupně přechází na interoperabilní aplikace a systémy, vyměňující si svá data prostřednictvím XML. Hnací silou jsou zejména systémy B2B, ale i internetové aplikace, integrující heterogenní a/nebo distribuovaná data.

Na výzvy XML reagují všichni význační dodavatelé databázových i aplikačních technologií, kteří je XML převážně integrují do stávajících systémů (IBM, Oracle, Software AG). Z globálního pohledu je ale de-facto nejkompletnějším pramenem nejnovějších technologií open-source vývoj, poskytující v různé kvalitě vše od základních XML middlewarových nástrojů (parsery, procesory, transformátory), XML databázových strojů, přes nástroje pro návrh webových uživatelských rozhraní až po hotové produkty (např. webové publikační systémy).

Problémem, ale zároveň největší tržní příležitostí zvláště pro menší a střední firmy, zůstává integrace těchto dílčích nástrojů dostupných jako open-source a budování průmyslových aplikací nad nimi, resp. jejich využití ve stávajících aplikacích.

Literatura

[BoLe] Bonifato, A., Lee, D.: Technical Survey of XML Schema and Query Languages, 2001

[EJB] Enterprise Java Beans Technology, http://java.sun.com/products/ejb

[Je01] Schematron, http://www.ascc.net/xml/resource/schematron/schematron.html

[KrŽe00] Král, J., Žemlička M., Autonomous Components, in Proc. of SOFSEM 2000, LCNS, Springer Verlag, 2000

[MS01] Microsoft XML Data Islands Howto, http://msdn.microsoft.com/xml/xmlguide/dataIslandHowTo.asp

[PokRi00] Pokorný, J., Richta, K.: XML a semistrukturovaná data, in Sborník DATASEM 2000, ISBN 80-210-2428-3

[UCM01] Unified Constraint Model, http://citeseer.nj.nec.com/385820.html

[XDuce00] XDuce, http://citeseer.nj.nec.com/279930.html

[XML98] eXtensible Markup Language, http://www.w3.org/XML

[XSLT99] XSL Transformations (XSLT), http://www.w3.org/TR/xslt.html

[Zv00] XSLT Tutorial, http://zvon.org/xxl/XSLTutorial/Books/Book1/index.html