



Pedagogická fakulta Jihočeské univerzity  
Katedra informatiky

# **E-business (E-commerce) a jazyk XML**

Bakalářská práce

Autor: Zdeněk Vlk

Vedoucí diplomové práce: PaedDr. Petr Pexa

**České Budějovice 2005**

*Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, pouze s použitím literatury a zdrojů uvedených v části Použité zdroje a v poznámkách pod čarou.*

Zdeněk Vlk

# Obsah

<b>ÚVOD</b> .....	<b>6</b>
<b>1 ELEKTRONICKÝ OBCHOD</b> .....	<b>8</b>
1.1 INTERNET .....	8
1.2 CO MŮŽEME ZAHRNOUT DO E-BUSINESSU .....	11
1.3 VYBRANÉ E-BUSINESS SYSTÉMY .....	14
1.3.1 Management vztahů se zákazníkem (CRM).....	14
1.3.2 Plánování firemních zdrojů (ERP).....	15
1.3.3 Management znalostí (KM).....	16
1.3.4 Management průběhu práce (WM).....	17
1.3.5 Management elektronických dokumentů (EDMS).....	17
<b>2 ELEKTRONICKÁ KOMERCE</b> .....	<b>19</b>
2.1 HISTORICKÝ VÝVOJ.....	19
2.2 DRUHY E-KOMERCE PODLE ZÚČASTNĚNÝCH STRAN.....	19
2.2.1 E-komerce typu B2C .....	20
2.2.2 E-komerce typu B2B .....	21
2.3 SYSTÉM ELEKTRONICKÉ VÝMĚNY DAT EDI .....	21
2.3.1 Standardizace EDI .....	22
<b>3 XML</b> .....	<b>24</b>
3.1 W3C .....	24
3.1.1 Schvalování standardů konsorciem W3C .....	24
3.2 SGML.....	25
3.3 HTML .....	26
3.4 ŽÁDNÉ PŘEDDEFINOVANÉ TAGY.....	27
3.5 PŘESNÁ SYNTAXE .....	28
3.6 PŘENOSITELNÝ DATOVÝ FORMÁT .....	28
3.7 SOUHRN VÝHOD A SPECIFICKÝCH RYSŮ JAZYKA XML.....	28
3.8 NEVÝHODY JAZYKA XML .....	30
3.9 PŘEHLED PŘÍBUZNÝCH TECHNOLOGIÍ XML.....	30
3.9.1 XML prostory jmen (XML namespaces).....	30
3.9.2 CSS (Cascading Style Sheets) .....	31
3.9.3 XSL (eXtensible Stylesheet Language).....	31
3.9.4 XLink (XML Linking Language).....	31
3.9.5 XPointer (XML Pointer Language).....	31
3.9.6 DTD (Document Type Definition) .....	32
3.9.7 XML Schémata .....	32
3.9.8 WML (Wireless Markup Language).....	32
3.9.9 DOM (Document Object Model).....	32
3.9.10 SAX (Simple API for XML) .....	32
3.9.11 JDOM (Java Document Object Model).....	33
3.9.12 JAXP (Java API for XML Processing).....	33
3.9.13 XML-RPC (XML – Remote Procedure Call).....	33

3.9.14 SOAP (Simple Object Access Protocol).....	33
3.9.15 XML Encryption.....	34
3.10 STRUČNÝ ÚVOD DO SYNTAXE JAZYKA XML.....	34
3.10.1 Hlavička XML dokumentu .....	34
3.10.2 Data v XML dokumentu .....	36
3.10.3 XML jmenné prostory (namespaces).....	40
3.11 STANOVENÍ SCHÉMA XML DOKUMENTU .....	41
3.11.1 DTD .....	41
3.11.2 Stručný úvod do XML Schémat .....	48
3.11.3 Skladiště DTD a XML schémat (Repositories).....	53
3.12 STYLOVÉ JAZYKY.....	53
3.12.1 Úvod do CSS .....	54
3.12.2 Úvod do stylového jazyka XSL .....	57
3.13 ZPRACOVÁNÍ XML DOKUMENTŮ .....	66
3.13.1 SAX .....	67
3.13.2 DOM .....	69
3.13.3 Stručný úvod do protokolu SOAP .....	71
3.14 PRAKTICKÁ UKÁZKA .....	74
<b>4 STANDARDY ELEKTRONICKÉ KOMERCE .....</b>	<b>75</b>
4.1 VYBRANÉ STANDARDY E-COMMERCE .....	75
4.1.1 cXML (Commerce XML).....	75
4.1.2 UBL(Universal Business Language) .....	76
4.1.3 ebXML (electronic business XML).....	78
4.1.4 Webové služby (Web Services).....	91
4.2 EBXML A WEBOVÉ SLUŽBY.....	93
<b>ZÁVĚR .....</b>	<b>96</b>
<b>SLOVNÍK POUŽITÝCH ZKRATEK.....</b>	<b>97</b>
<b>POUŽITÉ ZDROJE .....</b>	<b>101</b>

## Úvod

Pojmy „elektronický obchod“ a „jazyk XML“ již nejsou v oblasti informačních technologií nic nového pod sluncem. Ovšem elektronický obchod s využitím jazyka XML (eXtensible Markup Language) už takovou samozřejmostí není. Jazyk XML má mnoho předností, které se dají využít v této oblasti a právě těmito vlastnostmi jazyka se zabývá tato bakalářská práce.

Společně s jazykem XML vznikaly a stále vznikají mnohé další technologie, které doplňují a rozšiřují možnosti tohoto jazyka anebo představují nástroje ke zpracování a manipulaci s tímto jazykem. Mimo jiné, se také v této práci budu snažit uvést stručný přehled těchto technologií, protože mnohé z nich silně souvisejí s použitím XML v elektronickém obchodování.

Hlavním cílem této práce je analýza jazyka XML jako nástroje elektronického obchodování a výměny dat v elektronické komerci. Zdrojem pro tuto práci se stala domácí i zahraniční literatura a četné internetové zdroje (viz přehled literatury v závěru práce).

V úvodní kapitole vysvětlím pojem elektronického obchodu a důvody vzniku elektronického obchodování (v souvislosti se vznikem Internetu). Uvedu zde činnosti, které patří do elektronického obchodování.

V následující kapitole se pokusím zmapovat okruh činností e-businessu, které patří do podmnožiny e-businessu a to do oblasti e-commerce. Procesy e-komerce podrobněji rozvedu s důrazem na elektronickou výměnu obchodních dat a stručně popíšu starší technologii EDI (Electronic Data Interchange), která se stále používá, převážně ve větších firmách, v elektronickém obchodování (výměně obchodních dokumentů v elektronické podobě). Jelikož se v systémech EDI uplatňují určité standardy, zmíním některé organizace, které se touto standardizací zabývají a vyzdvihnu důležitost standardizace.

Dále se zaměřím na ty zvláštnosti jazyka XML (kapitola 3), které lze využít pro výměnu dat v e-komerci, a na výhody standardizace při globalizaci elektronického obchodování. Podrobně rozeberu ty části specifikace XML, které se využívají pro stanovení podoby obchodních dokumentů v XML a pro usnadnění zpracování těchto

dokumentů aplikacemi, které si je vyměňují. Popíše také jednu z technologií, která používá XML při posílání zpráv, a která se již používá v e-commerce.

V poslední kapitole této práce se pokusím nastínit některá schůdná řešení pro firmy, které by chtěli s elektronickým obchodováním začít nebo přejít z jiné technologie na technologii XML. Jedná se o několik standardů, které řeší otázky využití XML v e-commerce.

# 1 Elektronický obchod

V dnešní uspěchané době, kdy se zkracují vzdálenosti, kdy se člověk musí celý život učit novým a novým věcem, aby obstál v neúprosné konkurenci v boji o plný žaludek, nabývá potřeba informací na stále větší a větší důležitosti. Přesné, ale hlavně včas získané informace představují ohromnou výhodu pro jednotky operující v různých oblastech lidské činnosti. Jednou takovou oblastí je právě obchodování. Obchod byl důležitou součástí lidského života již od pradávna a prošel dlouhým, rozmanitým vývojem, a to především ve smyslu různorodosti nebo rozsahu obchodovaného zboží, zkracování doby transakce a zvyšování pohodlí obou zúčastněných stran.

Podniky dnešního typu, pokud se chtějí uplatnit na trhu, musejí podrobně analyzovat odvětví, ve kterém působí, musejí umět rychle a levně získat informace o konkurenci, o požadavcích spotřebitelů, o zákonných normách vztahujících se na toto odvětví a mnohé jiné. Na druhé straně spotřebitelé, ocení čím dál větší komfort při nakupování, vysokou rychlost nákupu a pokud možno nejnižší cenu. Firmy se musí snažit se svými zákazníky či dodavateli komunikovat rychle a faktury, objednávky nebo jiné doklady obchodního styku vyřizovat téměř okamžitě a ne se zpožděním několika dní.

## 1.1 Internet

Významným krokem k uskutečnění výše uvedených cílů, byl vznik Internetu. Zpočátku tuto tzv. světovou informační dálnici využívaly pouze armáda, výzkumná pracoviště a univerzity, a komerční využití sítě přicházelo v potaz až později. Počátkem 90. let, kdy byla poprvé aplikována technologie přenosu programů a postupně vytvořena dnes nejznámější část internetu – World Wide Web (WWW), se zákonitě Internet začal používat i ke komerčním účelům. V důsledku těchto tendencí, docházelo k mnoha změnám ve stylu obchodování. Například pro zrychlení transakcí nebo v důsledku tlaku zákazníků, dochází ke zkracování řetězce od výrobce ke spotřebiteli a to odbouráváním nepotřebných článků (dealerů, překupníků, ...), které nepřidávají žádnou hodnotu k danému výrobku nebo službě. To způsobilo mimo jiné také snížení nákladů na tyto „odpadlíky“ a snížení výdajů na větší zásoby [3].

Internet nabízí široké možnosti, jak efektivněji řídit chod podniku. Nyní si uvedeme přehled všeho, co v dobách bez Internetu, šlo jen s velkými obtížemi a dnes s tím firmy nemají skoro žádné problémy [3]:

- Své produkty, informace o nich a o firmě mohou nabízet – a prodávat – prostřednictvím webu, a to 24 hodin denně, 7 dní v týdnu.
- Mohou efektivněji a výhodněji nakupovat – vyhledat více dodavatelů, zaslat požadavky on-line, vyhledat nejlepší podmínky a mimořádné nabídky na on-line aukcích nebo na trhu s použitým zbožím.
- Po vyhledání dodavatelů, se s nimi mohou domluvit na způsobu komunikace a následně si přes Internet vyměňovat data a uskutečňovat tak obchodní činnost.
- Je možné urychlit systém objednávek, transakcí a plateb dodavatelům a snížit náklady.
- Mohou zkvalitnit najímání pracovníků, využijí-li on-line přehledů nabídek pracovních sil.
- Internet umožňuje zaměstnancům firmy získat lepší informace a možnost rekvalifikace.
- Podniky mají možnost založit intranet a usnadnit tak komunikaci mezi zaměstnanci a ústředím firmy a mezi zaměstnanci navzájem. Intranet lze využít ke sdílení informací, provozování školicích modulů, umístění kalendáře podnikových akcí atd.
- Mají možnost nabízet své zboží a obchodovat v globálním měřítku a rozšířit tak svou působnost na celý svět.
- Využitím Internetu se zvyšuje efektivita získávání dat o trzích, zákaznících, potenciálních zákaznících i konkurenčních firmách a lze provést přesnější a kvalitnější průzkum trhu.
- Mohou zasílat reklamy, katalogy a informace zákazníkům, kteří o ně požádají.

- Firmy mohou přizpůsobit nabídky, služby a prezentace informací potřebám a požadavkům jednotlivých zákazníků.
- S pomocí Internetu lze výrazně zlepšit logistický systém a provozní činnosti.

Přestože Internet poskytuje celou řadu známých a v dnešní informační době již nepostradatelných výhod, největší pozornost široké veřejnosti upoutal elektronický obchod, působící na Internetu jakožto oboustranný prodejně-nákupní kanál.

V důsledku toho vznikly a rychle se ujaly, dva nové termíny převzaté z angličtiny, kterými se budeme podrobněji zabývat. Jsou to **e-business** (elektronický obchod) a **e-commerce** (elektronická komerce). Co to tedy je e-business a co to je e-commerce? A je v nich vůbec nějaký rozdíl?

E-business je jakýkoli informační systém nebo aplikace, prostřednictvím které se uskutečňují obchodní transakce nebo jiné procesy související s obchodováním a řízením podnikových činností. Tyto systémy jsou dnes převážně založeny na webových technologiích.

E-commerci můžeme definovat, jako podmnožinu e-business aplikací, která se zaměřuje především na obchod typu B2B (*business-to-business*), B2C (*business-to-consumer*), C2C (*consumer-to-consumer*) a C2B (*consumer-to-business*). Jednotlivé pojmy si vysvětlíme v kapitolách „Elektronický obchod“ a „Elektronická komerce“.

Internet tedy představuje bránu do světa moderních technologií, které ovšem nelze začít využívat ze dne na den. Zaběhnutý systém, který tu existoval a ještě přetrvává, nelze naráz nahradit systémem, založeným výhradně na informačních a telekomunikačních technologiích. On-line obchodování rozhodně neznamená konec klasických prodejen a kamenné obchody budou ještě dlouhou dobu plnit svou roli.

K dosažení cílů plného využití elektronického obchodování je třeba mít k dispozici účinné prostředky sdílení, vyhledávání a výměny informací. Ukázalo se, že používané softwarové technologie spolu s moderními počítačovými sítěmi jako je Internet, potřebám elektronického obchodování nezcela vyhovují. Používají se proprietární formáty, se

kterými dovedou pracovat jen úzké okruhy aplikací, a výměna dat mezi informačními systémy jednotlivých společností je nákladná a zdaleka ne tak jednoduchá záležitost. (např. systémy EDI.

Další možností byl jazyk HTML (*HyperText Markup Language*), který však nedokázal uspokojit náročné úkoly, které dnešní elektronické obchodování a zejména pak elektronická komerce musí plnit. Příčiny neúspěchu HTML spočívaly v tom, že jazyk byl rozšiřován jednotlivými výrobci internetových prohlížečů, ve kterých jsou různé prvky jazyka podporovány jinak nebo nejsou podporovány vůbec. Ale hlavním důvodem odmítnutí jazyka, jakožto komplexního nástroje výměny dat bylo to, že jazyk se spíše zaměřuje na vzhledovou stránku než na logický obsah. Data představující předmět komunikace jsou schována v nepřehledném balastu formátovacích prvků.

Řešením tohoto problému se stal jazyk XML (*eXtensible Markup Language*), o kterém převážně pojednává tato práce.

## 1.2 Co můžeme zahrnout do e-businessu

Současný systém globální výměny informací je tvořen jednak sítí Internet, ale i řadou dalších sítí s omezeným uživatelským přístupem jako jsou sítě typu intranet (elektronická síť využívaná pro vnitropodnikovou komunikaci) a extranet – síť propojených intranetů – (síť pro komunikaci se zákazníky využívaná zejména pro výměnu informací, vyřizování objednávek, uzavírání kupních smluv a placení).

E-business je velmi široký pojem a můžeme ho chápat jako souhrn všech typů elektronické komunikace a obchodování, při kterých jsou využívány již zmíněné počítačové sítě.

Je mnoho způsobů jak klasifikovat aktivity e-businessu. Nejprve se tedy podíváme na e-business obecnějším pohledem. Při zkoumání oblastí použití *e-business aplikací* budeme brát v úvahu subjekty na obou koncích obchodní transakce a rozdělíme si je na dvě hlavní třídy:

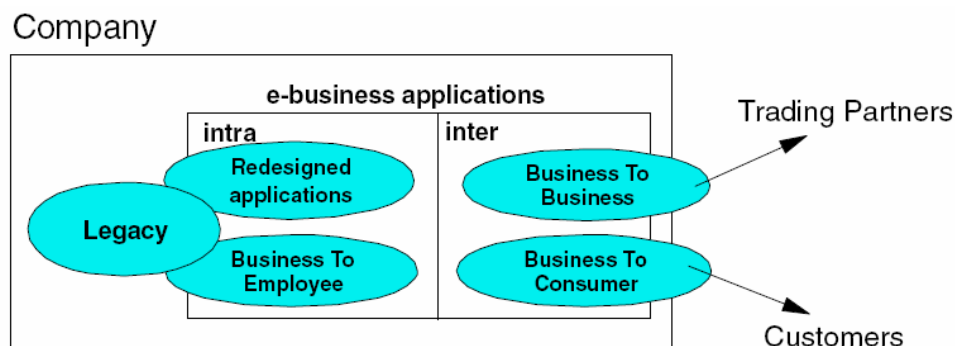
- **Intra-business**

- **Inter-business**

První třída, aplikace typu *Intra-business*, zahrnuje všechny e-business systémy, které firma nebo organizace používá uvnitř sebe sama, a které nepřesahují hranice podniku. Mohou to být například intranetové webové stránky pro zaměstnance.

Druhá třída, *Inter-business*, obsahuje všechny aplikace, které vyžadují jakýkoli druh interakce mezi společnostmi (firmou, podnikem) a nějakou externí entitou (zákazníkem, obchodním partnerem nebo finanční institucí). Jako příklad můžeme uvést aplikaci typu e-komerce, která modeluje nákupně-prodejní činnosti mezi firmou a spotřebitelem přes Internet. Je třeba ale dodat, že z infrastrukturního a implementačního úhlu pohledu, se tyto dvě třídy hodně prolínají a striktní hranice se jen těžko určuje. Následující obrázek ilustruje možné rozvržení e-business aplikací.

**Obrázek 1** Klasifikační schéma aplikací typu e-business



Pramen: Ennsner, Luis a kol. – Using XML for B2B and B2C Applications, IBM Redbooks, r. 2000

Přejdeme-li z obecného pohledu k více konkrétnějšímu, dají se e-business systémy rozdělit do těchto kategorií a podskupin.

1) Vnitropodnikové systémy:

- CRM – (*Customer Relationship Management*),
- ERP – (*Enterprise Resource Planning*),
- zaměstnanecké informační portály,

- KM – (*Knowledge Management*),
- WM – (*Workflow Management*),
- EDMS – (*Electronic Document Management Systems*),
- řízení lidských zdrojů,
- kontrola procesu,
- vnitropodnikové zpracování transakcí.

2) Komunikace a spolupráce mezi firmami:

- e-mail,
- voice mail (hlasová pošta),
- diskusní fóra,
- chat systémy,
- sdílení dat,
- spolupracující systémy.

3) Elektronická komerce – B2B, B2C, C2C a C2B (Inter-business):

- transfer elektronických cenných papírů nebo obchodních dokumentů,
- supply chain management (řízení dodavatelského řetězce),
- e-marketing,
- provádění on-line transakcí.

## 1.3 Vybrané e-business systémy

Nyní se seznámíme s některými e-business aplikacemi z předchozího přehledu a v následující kapitole probereme podrobněji kategorii třetí „Elektronická komerce“.

### 1.3.1 Management vztahů se zákazníkem (CRM)

CRM usiluje o co největší uspokojení potřeb zákazníků. Jednou z cest jak toho dosáhnout je, že firma přesvědčí své zákazníky o bezpečnosti a spolehlivosti procesů a procedur interaktivní komunikace s těmito zákazníky a tím by se měla dostat k zákazníkovi mnohem blíže, což evokuje kvalitnější a cílenější zákaznický servis. Úspěšná strategie implementace CRM je obvykle provázena integrací softwarového balíku upraveného pro podporu těchto procesů. Termín CRM může být použit jako popis určitého softwaru nebo firemní strategie orientované na potřeby zákazníka. Existují tři základní části architektury CRM:

- operativní – automatizace základních obchodních procesů (marketing, prodej, servis),
- analytická – podpora analýzy chování zákazníka, nákupních zvyků a shromažďování informací o spotřebitelích,
- kooperativní – zajišťuje kontakt se zákazníkem (telefon, email, fax, web, ...).

Příkladem použití této technologie mohou být telefonní centra, která používají CRM software k ukládání všech svých údajů o zákaznících. Když zákazník zavolá, systém je použit k získání a uložení informací příslušející danému zákazníkovi. A díky těmto informacím dokáže firma rychle a pohotově zákazníka obsloužit k všeobecné spokojenosti.

CRM řešení také dovoluje spotřebiteli provádět některé úkony vlastnoručně použitím různých komunikačních kanálů. Například je zákazníkovi umožněno zkontrolovat zůstatek na svém bankovním účtu pomocí svého telefonu díky technologii WAP (*Wireless Application Protocol*) a tím mu šetřit čas a poskytnout proklamované pohodlí.

### 1.3.2 Plánování firemních zdrojů (ERP)

ERP je manažerský informační systém, který integruje a automatizuje velké množství procesů souvisejících s produkčními činnostmi podniku. Typicky se jedná o výrobu, logistiku, distribuci, správu majetku, prodej, fakturaci, a účetnictví. Vzhledem k jeho širokému rozsahu působnosti, je třeba k vytvoření takovýchto komplexních systémů zaměstnat armádu analytiků a programátorů, a často se jedná o mnohamilionové projekty, které si mohou dovolit jen velké nadnárodní společnosti. Existují specializované firmy poskytující více či méně univerzální řešení, které si mohou firmy přizpůsobit svým konkrétním podmínkám. K výhodám ERP patří:

- nižší náklady na zásoby,
- nižší náklady na objednávky,
- nižší náklady na produkci,
- nižší náklady na vybavení,
- mnohem flexibilnější produkční prostředí,
- zvyšuje transparentnost procesu pro zákazníky.

Paradoxně právě kvůli své robustnosti a rozsáhlosti, mají tyto systémy i některé závažné nevýhody:

- jsou velmi drahé a nákladná je i údržba,
- některé jsou navíc poměrně složité a je třeba odborné obsluhy,
- systém je velice provázaný a pokud nastane problém v jednom oddělení, má to vliv i na ostatní části systému.

V případě integrace a konektivity s dodavatelskými systémy, se mohou vyskytnout i jiné dodatečné problémy:

- systém je velmi zranitelný a závisí na každém článku v dodavatelsko-odběratelském řetězci,
- po zavedení systému, může být pro jednu stranu změna ceny velmi nevýhodná,
- nejasnost hranic podniku a nedostatečné vymezení odpovědnosti či pravomocí nebo nelояálnost zaměstnanců, může způsobit také nemalé problémy,
- problém nekompatibility systémů všech zúčastněných stran,
- překážky ve sdílení citlivých interních informací, které jsou nutné k uskutečnění procesu.

Významnými dodavateli ERP jsou: SAP, J.D. Edwards, PeopleSoft, Lawson, Oracle Applications, Baan, WinStrom Software, Benefitt&Diatryma - česká firma.

Od roku 2000 je několik takovýchto systémů dostupných jako Open Source pod záštitou Open Source Licence. Stejně jako v případě komerčních dodavatelů, lze tyto systémy upravit k obrazu svému. Jsou to například ERP5<sup>1</sup> nebo nově vyvíjený OpenSCUM<sup>2</sup> (*Open Source Supply-Chain Unified Management Software*) založený na serveru Apache, databázi MySQL a skriptovacím jazyce PHP.

### 1.3.3 Management znalostí (KM)

KM je systém pro vytvoření, organizace, sdílení a tok znalostí uvnitř podniku. Jedná se o získávání a využití intelektuálního firemního kapitálu (standardů, obvyklých procesů a optimalizovaných postupů) a zkušeností ke zvýšení výkonnosti podniku. Existují softwarové balíky, které představují důležitou podporu KM. Tyto systémy se používají například pro správu informací o standardních postupech jednání s obchodním partnerem, pro uložení různých šablon a pravidel pro tvorbu dokumentů nebo pro správu údajů o zákazníkovi.

---

<sup>1</sup> [www.erp5.org/](http://www.erp5.org/)

<sup>2</sup> [www.openscum.org/](http://www.openscum.org/)

### 1.3.4 Management průběhu práce (WM)

WM představuje systém, který zajišťuje automatizaci a správu činnosti zvané workflow (tok práce). **Workflow** je problematika, zabývající se operativním aspektem pracovní procedury. Jak jsou úkoly podnikové činnosti strukturovány a propojeny, kdo je vykonává, pořadí činností, způsob vzájemné synchronizace, proud informací uvnitř podniku podporující jednotlivé úkoly a v neposlední řadě i důležitá kontrola jednotlivých činností.

Tento software zabezpečuje provádění toku práce, spolupracuje s účastníky procesu a v případě potřeby poskytuje použití dalších technologií a aplikací.

### 1.3.5 Management elektronických dokumentů (EDMS)

EDMS je systém, který se zaměřuje na ukládání běžných podnikových dokumentů v elektronické podobě. Má několik funkcí, které si stručně popíšeme.

První funkcí EDMS je **archivování**. Kvalitní a flexibilní archivační infrastruktura hraje podstatnou roli v efektivním EDMS řešení. Koncoví uživatelé potřebují být schopni jednoduše a konzistentně vkládat, uchovávat a indexovat dokumenty bez ohledu na rychlost, různost formátů či místo na disku.

Další hlavní funkcí EDMS je **administrace**. Robustní administrátorská činnost má úlohu pivota při implementaci EDMS. Jednou z mnoha činností administrátorů je definovat a kontrolovat autorizovaný přístup uživatelů do databáze dokumentů, zabezpečit systémové prostředí a zajistit harmonickou spolupráci EDMS se systémem WM.

Poslední neméně důležitou funkcí EDMS můžeme považovat **zpřístupnění**. Jestliže systém dokáže archivovat dokumenty, měl by zákonitě i dokumenty umět zpřístupnit jednotlivým uživatelům. Přístup k dokumentům znamená poskytnout uživateli potřebný dokument *vyhledat*, zajistit *dostupnost* dokumentu (např. z vzdálených kanceláří, v celé vnitropodnikové síti nebo prostřednictvím Internetu) a umožnit jejich *sdílení*.

Jak je vidno z předchozího výčtu, do elektronického obchodování lze zahrnout široké spektrum nejrůznějších aplikací, které spolu více či méně souvisejí. Jazyk XML, o němž je

převážně tato práce, by se určitě dal využít v mnoha těchto aplikacích a jistě se tak už dávno děje, ovšem já se zaměřím na oblast, ve které se tento jazyk stává čím dál populárnější a ve které přináší menší či větší výhody oproti jiným technologiím. Touto oblastí je právě **elektronická komerce** (B2B, B2C, C2C, C2B) zejména pak B2B a B2C. Uvedené zkratky si vysvětlíme v následující kapitole.

## 2 Elektronická komerce

Elektronická komerce chápána jako část elektronického obchodování, se skládá z nákupu, prodeje, marketingu a technické podpory produktů prostřednictvím počítačových sítí. V odvětví informačních technologií nazíráme na aplikaci v e-komerci jako na aplikaci zaměřenou převážně na obchodní transakce.

Alternativní definice říká, že e-komerce je obchodně-komerční komunikace a management za použití elektronických metod, jako jsou elektronická výměna dat (EDI) a automatizované systémy datových kolekcí.

### 2.1 Historický vývoj

Význam termínu „elektronická komerce“ se postupem času měnil. Původně, „e-komerce“ znamenala podporu elektronických komerčních transakcí, za použití technologie EDI k posílání obchodních dokumentů (objednávky, faktury) elektronicky.

Později se nazývaly tímto pojmem i další aktivity zvané „Web komerce“ – nákup zboží a služeb přes World Wide Web prostřednictvím zabezpečených serverů (HTTPS – speciální šifrovaný serverový protokol) s elektronickými košíky a službami elektronického placení kreditní kartou. V současné době se začínají prosazovat technologie výměny dat podporující jak EDI tak i Web komerci založené na jazyce XML.

### 2.2 Druhy e-komerce podle zúčastněných stran

- **B2B** (*business-to-business*) jak je patrné z anglické formulace zkratky, jedná se o obchodní vztah, realizovaný automatizovanými procesy a robustními softwarovými balíky, provozovaný převážně v prodejních a distribučních sítích a to mezi výrobcí, pobočkami, velkoobchody, distributory, dealery nebo obchodními zástupci. Základní rozdíl mezi B2B a B2C spočívá v tom, že prodávající (firma, distributor, dealer) nakupujícího předem zná. Obvykle jde o dlouhodobější vztah, který je ošetřen kupní smlouvou. Nejedná se tedy o klasické nakupování, ale

o uzavírání propojení mezi společnostmi. Příkladem tohoto typu komerce jsou třeba elektronická tržiště (*e-Market Place*, *B2B exchange*), na něž mají přístup jen registrovaní uživatelé. Software používaný pro provoz B2B je mnohem rozsáhlejší než se používá v obchodování typu B2C.

- **B2C** (*business-to-consumer*) obchod se specializuje na prodej zboží a služeb konečnému spotřebiteli. Výrobci a distributoři nabízejí své výrobky z větší části prostřednictvím Internetu kdy k přímému kontaktu prodávajícího a nakupujícího nemusí nikdy dojít a také většinou nedochází. B2C je již veřejnosti dobře známý a stále více využívaný systém. V některých případech je dokonce schopen klasický „kamenný“ obchod zcela nahradit.
- **C2C** (*consumer-to-consumer*) představuje vzájemný obchod mezi jednotlivými osobami, spotřebiteli. Jakýsi druh komerce ve formě směnného obchodu, burzy nebo blešího trhu. Za typický příklad elektronického C2C je považován internetový portál Ebay<sup>3</sup>.
- **C2B** (*consumer-to-business*) je sice méně obvyklý druh e-komerce, při níž individuální spotřebitelé nabízejí převážně své služby (práci) firmám a společnostem, ale jistě má svůj význam například na trhu pracovní síly.

### 2.2.1 E-komerce typu B2C

Spotřebitel účastníci se B2C e-komerce se nachází vně organizace a využívá systémy, známé také jako podnikové portály. Tyto portály odpovídají sadě Webových aplikací, zajišťujících konektivitu se zákazníkem, většinou prostřednictvím webového prohlížeče nebo s jeho informačním systémem, a poskytujících jednoduchou cestu k přizpůsobení nabídek a služeb jednotlivým zákazníkům a tak ulehčit jejich obchodní rozhodování.

---

<sup>3</sup> [www.ebay.com](http://www.ebay.com)

## 2.2.2 E-komerce typu B2B

Nejrozšířenější formou on-line obchodování B2B na Internetu jsou **elektronická tržiště** (ebay.com, amazon.com) a elektronická výměna dat EDI (Electronic Data Interchange). Princip elektronických tržišť spočívá ve vytvoření nabídkové databáze prodávající strany, na kterou má možnost kupující strana reagovat. Databáze je často tvořena z údajů obsažených ve vnitropodnikovém informačním systému prodávajícího a je z něho obvykle i automaticky aktualizována. V pracovanějších systémech je program schopen reagovat i na speciální přání na straně poptávky a to třeba poskytnutím množstevních slev, různých platebních podmínek apod. Takové systémy dokáží například vhodným způsobem zpracovat určitou objednávku zadáním požadavku do výroby, upravit obdržené údaje pro marketingovou činnost nebo navrhnout logistické řešení expedice objednaného zboží.

„Hlavní výhodou obchodování na elektronických tržištích je úspora času a nákladů spojených s vyhledáváním dodavatelů, i možnosti oslovit dodavatele na celosvětovém trhu. Některé prameny uvádějí, že by se v budoucnu mohlo prostřednictvím elektronických tržišť realizovat 30 % až 50 % světového mezifiremního obchodu.“ [1].

Okruh mezifiremního obchodování ovládl systém **EDI**, tj. systém elektronické výměny dat, který si popíšeme v následující části.

## 2.3 Systém elektronické výměny dat EDI

EDI systém můžeme definovat jako elektronickou výměnu strukturovaných dat mezi počítačovými systémy pomocí předem dohodnutých standardních zpráv a to s minimálními zásahy člověka. Přesto, že je systém EDI ve světě technologií jako je XML, Internet nebo Web relativně neznámý, je stále hnacím strojem 95% všech transakcí elektronické komerce na světě. **Elektronická výměna** dat znamená výměnu dokumentů elektronickou cestou, tedy za použití elektronických přenosů (online). V této komunikaci využíváme různé typy sítí – telefonní síť nebo datové síť. **Standardní zprávou** se zpravidla míní předem definovaný typ zprávy, v němž má každá položka své dopředu stanovené místo (standardní

zprávou může být například objednávka nebo faktura). **Strukturovanými daty** se rozumí data definovaná jistými syntaktickými pravidly.

### 2.3.1 Standardizace EDI

Z počátku si každá společnost či firma vytvořila svoje vlastní pravidla, která byla samozřejmě s pravidly ostatních firem nekompatibilní. S rostoucím počtem dodavatelů a obchodních partnerů (s různými typy systémů), vyvstala potřeba tyto pravidla pro tvorbu dat nějakým způsobem sjednotit či standardizovat. Syntaxe takto složených dat je zpravidla definována odvětvovými, národními a mezinárodními normami.

Dokumenty používané v systému EDI mají velmi podobné složení, jako jejich papírové protějšky. Například standard dokumentu EDI 940, se používá pro podání příkazu k odeslání zboží ze skladu maloobchodnímu odběrateli. Tento příkaz obsahuje adresu příjemce, adresu odesílatele, seznam čísel posílaných produktů (obvykle čárkových kódů) a jejich množství. Pokud se účastníci strany dohodnou, může dokument obsahovat i jiné dodatečné informace. Systémy EDI nacházejí uplatnění v různých oborech, a proto vznikají četné sady standardů, které popisují dokumenty specifické pro určitou oblast.

Existují tři sady EDI standard. UN/EDIFACT<sup>4</sup> (United Nations/Electronic Data Interchange for Administration, Commerce, and Transport) pod záštitou Spojených národů převládající kromě Severní Ameriky v celém světě. Další dva jsou pro Severní Ameriku: ANSI ASC X12<sup>5</sup> (American National Standards Institute Accredited Standards Committee X12) vyvinutý organizací ANSI a UCS<sup>6</sup> (Uniform Communication Standard).

Tyto systémy mohou zjednodušit, zlevnit a zefektivnit provoz ve firmě. Výhody používání EDI systémů:

- možnost použití různého hardware a software,
- nižší nutnost pracovního kapitálu a skladových zásob,

---

<sup>4</sup> [www.unece.org/trade/untdid](http://www.unece.org/trade/untdid)

<sup>5</sup> [www.x12.org](http://www.x12.org)

<sup>6</sup> [www.uc-council.org/ean\\_ucc\\_system/stnds\\_and\\_tech/ucs.html](http://www.uc-council.org/ean_ucc_system/stnds_and_tech/ucs.html)

- propojené řešení na obou stranách,
- redukce nákladů (méně papírování),
- snížení chybovosti (není nutné ruční zadávání dat),
- jednoduché zpracování velkého množství transakcí denně – rychlost.

Pokud firmy plně využijí možností těchto systémů, mohou se lidské zdroje dříve vázané na činnostech, které nyní zastává EDI systém, soustředit na procesy, které souvisejí převážně s podnikatelskou činností firmy. A nyní se podíváme na překážky, které musejí firmy překonávat, při implementaci EDI systémů. Nevýhody EDI systémů:

- vysoké náklady,
- dlouhá doba integrace do vnitropodnikových systémů a procesů,
- update systému z důvodu vývojových změn standardů,
- problematická synchronizace s podnikovými databázovými systémy.

Navíc v případě, že firma chce podnikat v mezinárodním měřítku, musí ve většinou podporovat standardy dva. Standard pro Severní Ameriku a standard pro zbytek světa, což opět zvyšuje náklady.

Jako elegantní řešení těchto problémů se nabízí jazyk XML. Tento jazyk otevírá malým a středním podnikům bránu do světa, kde až do teď operovaly pouze firmy a společnosti, které si mohli dovolit zavádět již zmíněné systémy EDI. Díky své rozšiřitelnosti, jednoduchosti a platformové nezávislosti poskytuje velmi flexibilní řešení pro specifické a měnící se prostředí v podnicích a firmách. Jazyk XML je volně šířitelný, jeho specifikaci si může každý zdarma prohlédnout na serveru konsorcia W3C, a jeho implementace je mnohem levnější než implementace klasických EDI systémů. XML a standardy EDI jsou hlavní technologie, které se dnes používají pro tvorbu aplikací typu B2B a i když jazyk XML poskytuje levnější a efektivnější řešení, nejrozšířenější stále zůstávají systémy EDI.

## 3 XML

Proč byl jazyk XML vyvinut? Co vedlo tvůrce k vytvoření jazyka? Jaké výhody skýtá použití XML? Jak je konkrétně využívá v elektronickém obchodování? Převážně na tyto otázky se budu snažit odpovědět v následujících kapitolách této práce.

XML<sup>7</sup> (eXtensible Markup Language) je staronový značkovací jazyk vyvinutý konsorciem W3C (World Wide Web Consortium)<sup>8</sup> za účelem překonat omezení jazyka HTML, ale zároveň zachovat jeho jednoduchost.

### 3.1 W3C

W3C je mezinárodní společenství různých organizací, veřejných subjektů a univerzit za účelem vývoje standardů pro World Wide Web ve formě tzv. „W3C doporučení“. W3C vydalo první verzi XML 1.0 v únoru 1998 a poslední verze jazyka XML 1.1 byla uvedena v únoru 2004. Proč se jedná pouze o doporučení a nikoli o standardy v pravém slova smyslu? Instituce se tím brání proti možnosti vzniku soudních sporů o porušování antimonopolních zákonů.

#### 3.1.1 Schvalování standardů konsorciem W3C<sup>9</sup>

Vznik W3C doporučení zahrnuje sedm kroků, kterými musí každý návrh standardu projít:

- **Návrh** (W3C Submission) – členové konsorcia mají možnost podat návrh nového Web standardu, který je konsorciem posouzen (zda spadá do působnosti konsorcia) a přípuštěn k dalšímu zkoumání.
- **Záznam** (W3C Note) – je popis návrhu, zveřejněný na stránkách konsorcia, ale zdaleka to ještě neznamena, že W3C začne na návrhu pracovat.

<sup>7</sup> [www.w3.org/TR/REC-xml/](http://www.w3.org/TR/REC-xml/)

<sup>8</sup> [www.w3c.org](http://www.w3c.org)

<sup>9</sup> [www.w3schools.com/w3c/w3c\\_process.asp](http://www.w3schools.com/w3c/w3c_process.asp)

- **Pracovní skupina** (W3C Working Group) – v případě že je návrh konsorciem schválen, je sestavena pracovní skupina složená s členů konsorcia a jiných zainteresovaných stran, která si určí časový plán a vydává pracovní náčrty.
- **Pracovní náčrt** (W3C Working Draft) – je běžně zveřejňován na stránkách W3C, pro posouzení veřejnosti. Tento nástin práce by však ještě neměl být zaváděn do praxe, poněvadž může být kdykoli změněn či dokonce odstraněn.
- **Kandidátské doporučení** (W3C Candidate Recommendation) – některé specifikace jsou mnohem rozsáhlejší a komplexnější než jiné a vyžadují více času atestování, proto jsou tyto specifikace publikovány jako kandidátské doporučení. Ovšem stále je to pracovní verze a nelze ji brát jako konečné řešení, pořád může být totiž úplně odstraněna.
- **Plánované doporučení** (W3C Proposed Recommendation) – tento stav specifikace je finální fáze činnosti pracovní skupiny. Sice je to stále pracovní verze, ale plánované doporučení je už hodně blízko konečné verzi doporučení.
- **Doporučení** (W3C Recommendation) – je schváleno členy W3C a představuje stabilní, konečný dokument, který může být použit jako referenční materiál.

## 3.2 SGML

XML nevzniklo jen tak z ničeho nic. Jazyk s podobnou filosofií, spočívající v upřednostnění označení *významu* částí dokumentu nad jeho *vzhledem* tu už byl mnohem dříve. Byl to jazyk SGML (Standard Generalized Markup Language), který byl definován v ISO<sup>10</sup> normě 8879 již v roce 1986.

Tento jazyk umožňuje definování vlastních sad značek a jejich vzájemných vztahů prostřednictvím tzv. *definice typu dokumentu* (DTD) a to s velmi vysokou mírou obecnosti a komplexnosti. Bohužel právě kvůli univerzálnosti a veliké složitosti se nedal plně využít v praxi.

---

<sup>10</sup> [www.iso.org](http://www.iso.org)

HTML a XML jsou oba odvozeny ze SGML, avšak každý jiným způsobem. Zatímco HTML je jedna z aplikací SGML (sada značek) určená příslušným DTD<sup>11</sup>, XML je specifická podmnožina SGML sestavená tak, aby se dala softwarově zpracovávat a analyzovat mnohem jednodušeji. Další značkovací jazyk vyvinut jako aplikace SGML je dnes již známý DocBook, určený zejména pro tvorbu počítačové dokumentace. DocBook je k dispozici i jako aplikace XML.

### 3.3 HTML

Přestože XML v mnoha ohledech jazyk HTML převyšuje, jazyk HTML stále ještě není mrtvým jazykem a k účelu zobrazování a prezentaci jednoduchých statických internetových stránek je plně dostačující. Ovšem požadavkům náročnějších uživatelů Internetu již zdaleka nestačí a je třeba ho doplňovat modernějšími, dynamičtějšími a flexibilnějšími technologiemi jako jsou ASP, PHP, JavaScript, Java, CSS, PNG, Flash, MP3, WAV a mnohé jiné, z nichž některé byly také vyvinuty konsorciem W3C.

Z dnešního Webového prostředí se stává dynamický, multimediální kolos, který je schopen plnit čím dál více našich nejtajnějších snů. Od elektronického bankovníctví, online obchodů přes diskusní fóra až po flashové animace či dokonce virtuální světy (téměř k nerozeznání od reality).

Zpočátku se konsorcium snažilo vyřešit omezenost jazyka HTML přidáváním dalších a dalších potřebných i méně potřebných tagů (značek). Poslední verze HTML 4.01<sup>12</sup> vydaná v prosinci 1999 skýtá bez mála 100 tagů, což už představuje poměrně rozsáhlou sadu. S čím dál větším počtem HTML značek byl i vývoj prohlížečů, náročnější, složitější. Hardwarové nároky na přenos takovýchto HTML stránek byl (a stále je) vyšší. Ovšem navzdory poměrně velikému počtu značek, je jich pořád málo. Různorodost odvětví lidské činnosti staví jazyk HTML do nezáviděníhodné pozice. Například e-komerce by například potřebovala přidat tagy jako jsou cena, adresa, popis, název, IČO atd. Jiné oblasti by

---

<sup>11</sup> Identifikátor DTD pro finální verzi HTML 4.0 vypadá například takto: `-//W3C//DTD HTML 4.0 Final//EN`.

<sup>12</sup> Přesněji za poslední verzi HTML se považuje XHTML 1.0, což je přeformulovaná verze HTML 4.01 do XML.

vyžadovali značky specifické právě pro svůj okruh činností a tak bychom mohli pokračovat do nekonečna. Rozšiřováním jazyka tedy cesta nevede.

Jak tedy vyřešit tyto protichůdné postoje, kdy pro člověka jako uživatele by bylo potřeba tagů více, zatímco vývojář by ocenil, kdyby bylo tagů méně? Tento problém řeší XML tím, že ve své specifikaci *žádné předdefinované tagy* nemá a zároveň jeho *syntaxe je velice přísná*.

### 3.4 Žádné předdefinované tagy

V době, kdy jsem se začínal učit jazyk XML, mne příjemně překvapila skutečnost, že XML nemá žádné předdefinované tagy. A to z mnoha důvodů. Jednak proto, že jsem se nemusel učit nové a nové značky, jako tomu bylo u jazyka HTML, a jednak proto, že toto pojetí značkovacího jazyka je velice otevřené a nabízí velikou volnost při tvorbě dokumentů. Ono to tak úplně není, ale k tomu se ještě dostaneme.

Autor dokumentů si může zvolit své vlastní elementy, které chce a hlavně potřebuje použít ke své práci. Jak již bylo uvedeno výše, podnik provozující elektronické obchodování má nyní možnost si pro své dokumenty zvolit značky, které budou vyhovovat právě jeho předmětu obchodní činnosti. Zejména pro e-komerci, jak je vidět z následující ukázky, se používají elementy popisující například zboží, služby, počty kusů výrobků, adresy zúčastněných stran atp.

```
<objednávka datum="8.3.2005">
  <adresa_dodavatele>
    <jméno>Zeos s.r.o.</jméno>
    ...
  </adresa_dodavatele>
  ...
  <zboží>
    <číslo_položky>89</číslo_položky>
    <název>Dřevotřísková deska</název>
    <počet_kusů>25</počet_kusů>
  </zboží/>
  ...
</objednávka>
```

V dalších částech této práce si stručně vysvětlíme, jak byla ošetřena tato volnost a jak se s tím vypořádávají aplikace a webové prohlížeče.

## 3.5 Přesná syntaxe

Toto je dobrá zpráva hlavně pro vývojáře prohlížečů a aplikací, které budou dokumenty napsané v XML zpracovávat. Díky striktní syntaxi je mnohem jednodušší napsat aplikaci, která bude přesně vědět co má čekat a nebude zde prostor pro nejednoznačnosti jako to bylo v případě jazyka HTML. Takovéto aplikace by měli být menší, mnohem rychlejší a měli by je být schopny provozovat i méně výkonné stanice (PDA, PocketPC nebo mobilní telefony). V současné době se tak děje například ve formě wapových stránek napsaných v jazyce WML.<sup>13</sup>

## 3.6 Přenositelný datový formát

Od samého začátku se jazyk XML snaží být přenositelným, na platformě a jazyce nezávislým. Jak je toho docíleno? XML splňuje dvě důležité věci: je to jednoduchý čitelný text, tudíž si ho mohou operačními systémy snadno mezi sebou vyměňovat, a má podobu standardu konsorcia W3C! Druhá skutečnost je velmi důležitá pro aplikace (napsané v různých programovacích jazycích). Tyto programy (znalé specifikace XML) totiž vědí zcela přesně jak zpracovat XML dokument, který mají na svém vstupu.

## 3.7 Souhrn výhod a specifických rysů jazyka XML

- **Jednoduchost** – data uchovaná v dokumentech XML jsou čitelná a srozumitelná a mohou být jednoduše zpracována strojově.
- **Přenositelnost** – přenositelný datový formát.
- **Otevřený formát** – XML není majetkem žádného komerčního subjektu. Je to volný standard, který byl vyvinut konsorciem W3C.
- **Strukturovanost** – XML se snaží zachytit strukturu informací a jejich vzájemné vztahy, nezávisle na způsobu jejich zobrazení.

---

<sup>13</sup> WML dokumenty jsou XML dokumenty validované (platné) podle DTD specifikace WAP (Wireless Application Protocol).

- **Rozšiřitelnost** – možnost rozšíření o další elementy, pokud jsou potřeba.
- **Samopopisný** – v tradičních databázích se data popisují pomocí tabulek či schémat, ale dokumenty už tyto informace o datech (metadata) obsahují ve formě elementů ohraničujících tyto data a jejich atributů.
- **Strojově čitelné kontextové informace** – Struktura tagů a jejich atributů poskytuje kontextové informace, které mohou být použity k interpretaci významu obsahu dokumentu. Tato vlastnost mimo jiné otevírá bránu k mnohem efektivnější a kvalitnější práci vyhledávacích enginů, což u holého textu nebo dokumentů napsaných v HTML je velice obtížné.
- **Podpora tvorby mezinárodních dokumentů (Unicode, ISO 10646)**<sup>14</sup> – XML podporuje znakovou sadu ISO 10646 a Unicode, což je důležité při tvorbě mezinárodních dokumentů a aplikací.
- **Výhody porovnání a seskupení dat** – stromová struktura XML dokumentů dovoluje efektivní porovnávání a seskupování jednotlivých dokumentů element po elementu.
- **Vkládání různorodých datových typů** – XML dokumenty mohou obsahovat jakýkoli možný datový typ – od multimediálních datových typů (video, zvuk, obrázek) až po aktivní prvky (ActiveX, Java aplety).
- **Poskytuje „one server“ pohled pro rozdělená data** – XML dokumenty mohou být složeny z vnořených elementů, které jsou roztroušeny na různých vzdálených serverech. I celý World Wide Web může být viděn jako jedna obrovská XML databáze.
- **Rychlé přijetí XML v různých odvětvích** – mnohé softwarové firmy jako Microsoft, Sun, Netscape, IBM už dávno zahrnuly podporu XML do svých

---

<sup>14</sup> ISO organizace a Unicode konsorcium se v roce 1991 rozhodli, že vytvoří společně univerzální standard pro kódování multi-jazykových textů. ISO ho označuje jako *ISO 10646* a Unicode konsorcium jako *Unicode*. Ačkoli jsou tyto dva standardy synchronizovány, nejsou úplně stejné. Unicode obsahuje navíc speciální dodatky a omezení, která zajišťují nezávislost znaků na platformách a aplikacích.

produktů a nejenom XML, také ostatní technologie související s XML mají své nezanedbatelné místo.

## 3.8 Nevýhody jazyka XML

- **Binární formát vs. XML dokument** – XML soubor s daty je skoro vždy větší než srovnatelná data v binárním formátu (zabírají na disku více místa). Řešením mohou být kompresní programy; při přenosu po síti dokáže protokol HTTP 1.1 XML data také zapakovat, načež zatížení linky je srovnatelné s binárními soubory.
- **XML a binární data** – jazyk XML není vhodný pro ukládání binárních dat (binárního kódu programů, obrázků nebo videa). Často však není nutné, aby tyto soubory byly čitelné pro člověka, proto nemusíme tyto data nutně převádět do XML.

## 3.9 Přehled příbuzných technologií XML

Společně s XML byly vyvinuty mnohé další standardy a technologie (nejen z dílny konsorcia W3C) doplňující a rozšiřující výhody jazyka XML. Není náplní této práce uvést všechny (ono by to ani dost dobře nešlo), ale uvedeme si ty (podle mého názoru) nejnámější nebo nejpoužívanější. Několika z nich, v souvislosti s elektronickým obchodováním, se budeme v dalších kapitolách věnovat podrobněji.

### 3.9.1 XML prostory jmen (XML namespaces)

Mechanismus prostorů jmen, připomínající jmenné prostory z běžných programovacích jazyků, se používá hlavně k rozeznávání názvů elementů a atributů mezi různými XML slovníky (sadami značek), čímž se předchází možné kolizi jmen. Prostory jmen se mohou použít také k jakémusi odlišení elementů a atributů příslušejících určité XML aplikaci, tak aby je software mohl snadněji rozeznat.

### 3.9.2 CSS (Cascading Style Sheets)

CSS je další z mnoha doporučení vyvinuto konsorciem W3C. Je to stylový jazyk používaný pro popis vzhledu strukturovaných dokumentů napsaných v značkovacích jazycích jako HTML, XHTML a XML. Existují i starší stylové jazyky, jako například DSSSL, který se používá pro dokumenty SGML. Je velmi mocný, ale stejně jako SGML, je moc rozsáhlý, komplexní a silně nepraktický pro Web.

### 3.9.3 XSL (eXtensible Stylesheet Language)

XSL stylový jazyk tvoří jakousi rodinu doporučení, která definují transformaci a prezentaci dokumentů. Obsahuje tři části:

- **XSLT** (XSL Transformations) – jazyk pro *transformaci* XML dokumentů,
- **XPath** (XML Path Language) – výrazový jazyk užívaný v XSLT jako přístupová cesta k částem XML dokumentu (XPath se také používá ve specifikaci XLink),
- **XSL-FO** (XSL Formatting Objects) – umožňuje formát objektů, které vzniknou transformací XML dokumentu, do podoby výsledného dokumentu.

### 3.9.4 XLink (XML Linking Language)<sup>15</sup>

XLink je XML značkovací jazyk, který zastřešuje tvorbu odkazů mezi XML dokumenty, narozdíl od odkazů v HTML poskytuje mnohem komplexnější odkazovací strukturu. Jednak lze tvořit odkazy mezi více než dvěma zdroji, doplnit k odkazům metadata nebo umístit odkazy mimo odkazované zdroje.

### 3.9.5 XPointer (XML Pointer Language)<sup>16</sup>

XPointer doplňuje jazyk XLink o odkazy uvnitř dokumentu.

---

<sup>15</sup> [www.w3.org/TR/xlink/](http://www.w3.org/TR/xlink/)

<sup>16</sup> XPointer pracovní návrh se nachází na adrese [www.w3.org/TR/xptr/](http://www.w3.org/TR/xptr/).

### 3.9.6 DTD (Document Type Definition)

DTD je definicí typu dokumentu. Stanovení sady značek, počtů výskytů a jiných omezení. Tato technologie časově předcházela jazyku XML a byla zděděna z jazyka SGML spolu s téměř původní syntaxí. Bohužel možnosti DTD jsou velice omezené, například chybí schopnost určení datových typů.

### 3.9.7 XML Schémata

DTD je primárně zaměřeno na popis jak mají být elementy v dokumentu uspořádány, ale už moc neříká o datovém obsahu dokumentu. Ačkoli atributy mohou být deklarovány jako různé typy (ID, IDREF, výčet), u elementů to nejde. Tyto a další problémy se snaží řešit XML schémata, která jsou navíc sama XML dokumenty.

### 3.9.8 WML (Wireless Markup Language)<sup>17</sup>

WML je značkovací jazyk pro mobilní telefony a jiná bezdrátová zařízení.

### 3.9.9 DOM (Document Object Model)

Standard vyvinutý opět konsorciem W3C, který si klade za cíl být platformově a jazykově nezávislé rozhraní, které poskytuje programům a skriptům dynamicky přistupovat k dokumentům a měnit jejich obsah a strukturu. K těmto úkolům využívá stromovou strukturu odvozenou z příslušného XML dokumentu.

### 3.9.10 SAX (Simple API for XML)

Toto aplikační rozhraní, jež není dílem konsorcia W3C, je založeno na událostech, které jsou vyvolány postupným procházením XML dokumentu.

---

<sup>17</sup> [www.openmobilealliance.org/tech/affiliates/wap/wapindex.html](http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html)

### 3.9.11 JDOM (Java Document Object Model)<sup>18</sup>

JDOM je **open source**<sup>19</sup> knihovna, která si klade za cíl, zjednodušit manipulaci s XML dokumenty v programovacím jazyce Java. Ačkoli je JDOM model podobný technologii DOM od konsorcia W3C, není na DOM postaven a jde svou vlastní cestou. Hlavní rozdíl mezi DOM a JDOM spočívá v tom, že DOM byl vyvinut tak, aby byl jazykově nezávislý, naproti tomu JDOM je výhradně postaven na programovacím jazyku Java.

### 3.9.12 JAXP (Java API for XML Processing)<sup>20</sup>

JAXP je, jak ukazuje název, aplikační rozhraní, které poskytuje nástroje k vytváření XML dokumentů, parsování pomocí technologií SAX nebo DOM a transformování pomocí XSLT. Uplatňuje nezávislost na implementaci parseru<sup>21</sup> (možnost použití parserů od různých výrobců).

### 3.9.13 XML-RPC (XML – Remote Procedure Call)<sup>22</sup>

XML-RPC je sada implementací, která umožňuje aplikacím běžícím na různých operačních systémech, volat procedury prostřednictvím Internetu. K tomuto účelu používá protokol HTTP (transportní vrstva) a jazyk XML (kódovací vrstva).

### 3.9.14 SOAP (Simple Object Access Protocol)

SOAP je protokol pro zasílání XML zpráv, který umožňuje tyto zprávy posílat jednosměrně mezi odesílající a přijímací aplikací. I když je tato technologie pouze jednosměrná, běžná komunikace peer-to-peer se pomocí ní provozuje celkem bez problémů.

---

<sup>18</sup> [www.jdom.org/](http://www.jdom.org/)

<sup>19</sup> [www.opensource.org/](http://www.opensource.org/)

<sup>20</sup> [java.sun.com/xml/jaxp/index.jsp](http://java.sun.com/xml/jaxp/index.jsp)

<sup>21</sup> Parser je speciální program, který je přímo určen ke zpracování XML dokumentů.

<sup>22</sup> [www.xmlrpc.org/](http://www.xmlrpc.org/)

### 3.9.15 XML Encryption<sup>23</sup>

Tento standard je v současné době zatím ve fázi (fáze W3C doporučení viz. výše), kdy byla ustanovena pracovní skupina, která má za úkol vyvinout způsob jak zašifrovat a dešifrovat digitální data (včetně XML dokumentů nebo jeho částí). Při tom pro samotné šifrování obsahu a specifické informace (klíče) k dešifrování by měla být použita syntaxe XML.

## 3.10 Stručný úvod do syntaxe jazyka XML

V této kapitole si uvedeme několik důležitých syntaktických pravidel jazyka XML. XML dokument můžeme rozdělit do dvou částí: *hlavička* a *data*. Hlavička podává užitečné informace XML parserům a jiným aplikacím o tom, jak mají daný dokument zpracovávat. Nyní si ukážeme jednotlivé části dokumentu na příkladu.

```
<?xml version="1.0" encoding="ISO-8859-2" standalone="no"?>
<!DOCTYPE sestava SYSTEM "typy/sestava.dtd">
<!-- Počítačová sestava -->
<sestava xmlns="http://www.vlkcomputers.cz/sestavy"
  xmlns:dod="http://www.velkoobchod_comp.cz">
  <titulek dod:serie="45TFG">Počítačové díly</titulek>
  <díly>
    <zákl_deska>
      <název>ASUS A7N8X Deluxe</název/>
      <čipset>nVidia NForce4</čipset>
    </zákl_deska>
    ...
  </díly>
  <dod:copyright>&vlkcomputers_Copyright;</dod:copyright>
</sestava>
```

### 3.10.1 Hlavička XML dokumentu

Hlavička obsahuje tzv. *instrukci pro zpracování* ohraničenou řetězcí „<?xml“ a „?>“. Instrukce pro zpracování se používají pro zvláštní sdělení zpracovávající aplikaci.

```
<?xml version="1.0" encoding="ISO-8859-2" standalone="no"?>
```

<sup>23</sup> [www.w3.org/Encryption/2001/](http://www.w3.org/Encryption/2001/)

Tato instrukce představuje deklaraci XML dokumentu a může obsahovat tři parametry:

- `version` – verze specifikace jazyka XML; v současné době může nabývat hodnot 1.0 a 1.1, ovšem většinou XML procesorů je zatím podporována pouze verze 1.0,
- `encoding`<sup>24</sup> – udávající kódování dokumentu,
- `standalone` – má buď hodnotu „yes“ nebo „no“. Pokud má hodnotu „no“, znamená to, že existuje ještě nějaký externí zdroj, který je nutný proto, aby parser mohl korektně zpracovat XML dokument.

### Hlavní rozdíly mezi XML 1.0 a XML 1.1

- XML 1.1 podporuje novější verzi sady Unicode.
- Verze 1.1 uvolňuje některá pravidla pro tvorbu jmen elementů a atributů (povoluje užití více znaků sady Unicode a zohledňuje případné budoucí rozšíření této sady).
- V XML 1.1 existuje více znaků k ukončení řádku (NEL 0x85 definovaný společností IBM a také znak 0x2028 definovaný přímo standardem Unicode).
- Ve verzi 1.1 je deklarace dokumentu povinná, zatímco ve verzi 1.0 ji můžeme vynechat.

### Komentář

Komentáře v XML jsou velmi podobné komentářům v HTML, ale podléhají přísnějším pravidlům. Například komentář XML se nemůže objevit v uvnitř značky. Toto není korektní umístění komentáře:

```
<název <!-- Počítačová sestava --> >obsah elementu</název>
```

---

<sup>24</sup> XML podporuje rozsáhlé kódování ISO 10646 (Unicode), které uchovává jeden znak do 32 bitů, nebo jeho odlehčené verze UTF-8 (8 bitů na znak) a UTF-16 (16 bitů na znak). Konsorcium W3C vyžaduje u XML procesorů minimální podporu UTF-8 nebo UTF-16. XML procesory často podporují i mnohé jiné kódovací sady (ISO-8859-2, Unicode, windows-1250, ...).

## Deklarace DTD

Další součástí hlavičky XML dokumentu může být deklarace typu dokumentu, která se uvozuje DTD značkou „<!“, následovanou klíčovým slovem DOCTYPE, po kterém následuje kořenový element, klíčové slovo SYSTEM nebo PUBLIC a cesta k souboru nebo veřejný identifikátor a vše uzavírá znak „>“. Cesta k souboru a veřejný identifikátor se často používají oba současně, protože v případě, že parser nerozezná příslušný identifikátor, použije cestu pro nalezení souboru s definicí DTD. Podrobnější popis definice a deklarace DTD uvedu v části „Stanovení schéma XML dokumentu“. Deklarace externího DTD:

```
<!DOCTYPE sestava SYSTEM "typy/sestava.dtd">
```

Zde je definice typu dokumentu uložena lokálně ve složce *typy/* v souboru *sestava.dtd*. Klíčové slovo SYSTEM se používá v případě použití *cesty k souboru* DTD. Cesta může být buď relativní, absolutní nebo URI. Pojem URI si vysvětlíme v části „XML jmenné prostory“. Ve výše uvedeném příkladu je použita relativní cesta k souboru. Další možností je použití *veřejného identifikátoru*. To znamená, že W3C nebo jiná organizace definuje standardní DTD, které je asociováno s určitým veřejným identifikátorem. K tomuto účelu se používá klíčové slovo PUBLIC.

```
<!DOCTYPE sestava PUBLIC "-//VLKCOMPUTERS//DTD Sestava 1.1//CS"
"typy/sestava.dtd">
```

### 3.10.2 Data v XML dokumentu

Po hlavičce dokumentu následuje část datová, která obsahuje elementy, atributy a ukládané informace. Tato část dokumentu je ohraničena kořenovým elementem, který musí obsahovat každý XML dokument a může být pouze jeden. Roli kořenového elementu hraje v mém předešlém příkladu element *sestava*.

```
<sestava xmlns="http://www.vlkcomputers.cz/sestavy"
xmlns:dod="http://www.velkoobchod_comp.cz">
</sestava>
```

## Elementy

Tvorba elementů v XML dokumentu podléhá určitým pravidlům, které se ve verzích XML 1.0 a XML 1.1 malinko různí (zejména v tvorbě jmen elementů).

- Elementy v XML dokumentu se nesmějí překrývat.
- Každý element musí mít počáteční značku `<název_elementu>` a koncovou značku `</název_elementu>`. Výjimku tvoří prázdný element, který obsahuje obě značky v jedné `<obrazek zdroj="obr/obrazek.img"/>` (tyto elementy se většinou používají pro vkládání netextových dat).
- V názvech elementů se rozlišují velké a malé znaky.
- Název počáteční značky musí být stejný jako název konečné značky.

Ve verzi XML 1.0 mohou jména elementů začínat písmenem, podtržítkem nebo dvojtečkou a další znaky mohou být písmena, číslice, podtržítka, pomlčky a tečky (bez mezer a jiných bílých znaků).

Ve verzi XML 1.1 jsou jména tvořena více méně podobně, ovšem nově se uplatňuje pravidlo: „Co není zakázáno, je povoleno“. Filozofie tohoto pravidla vychází z toho, že znaková sada Unicode se bude dále rozvíjet a rozšiřovat, proto XML jde cestou povolení znaků, které ještě nejsou v Unicode zavedeny.

## Atributy

Uvnitř značek elementů, mohou být vloženy atributy, které jsou podobné atributům v HTML, ale jsou podřízeny přísnějším pravidlům. Hodnota atributů musí být uvozena buď jednoduchými nebo složenými uvozovkami. Jména parametrů se tvoří stejně jako jména elementů. Po jménu následuje rovnítko a hodnota parametru.

```
<zákl_deska ser_číslo="RT567" název="ASUS A7N8X Deluxe">  
  <čipset jméno="nVidia Nforce &číslo;"></čipset>  
</zákl_deska>
```

Neexistují jednoznačná pravidla kdy použít elementy a kdy atributy. Elementy jsou pravděpodobně vhodnější pro rozsáhlejší nebo výčtové hodnoty a atributy se většinou používají pro krátké doplňující informace. Někdy je ovšem obtížné se rozhodnout zda použít element či atribut a to co je lepší se ukáže až v praxi.

## Entity

Entity velice usnadňují život programátorům XML, protože zjednodušují psaní a úpravu XML dokumentů. Pomocí entit lze například vkládat znaky, které by parser označil za syntaktickou chybu (&, <, >, ...).

Entity se rozdělují do několika kategorií: *obecné* nebo *parametrické*, *interní* nebo *externí* a *textové* nebo *binární*:

- **Obecná entita** – obsahuje textová data, XML text nebo netextová data. Tyto entity se vkládají do kořenového elementu dokumentu.
- **Parametrická entita** – obsahuje text a vkládá se do DTD dokumentu.
- **Interní entita** – je ohraničený řetězec a je umístěna uvnitř dokumentu.
- **Externí entita** – jedná se o externí soubor, který je deklarován jako entita v DTD.
- **Textová entita** – obsahuje XML text.
- **Binární entita** – může obsahovat text nebo binární data (obrázek, zvuk, video, ...).

Tabulka 1 Rozdělení entit

	TEXTOVÁ	BINÁRNÍ	INTERNÍ	EXTERNÍ
Obecná entita	X		X	
	X			X
		X		X
Parametrická entita	X		X	
	X			X

Pramen: J. Young, Michael – XML krok za krokem, 2002

Předcházející tabulka ukazuje různé druhy obecných i parametrických entit. Obecné entity se tvoří pomocí počátečního znaku „&“ a koncového „;“, parametrické podobně, ale místo ampersandu se zapisuje znak procenta „%“. A nyní si ukážeme použití entit na příkladech.

#### Obecná textová-interní, textová-externí a binární-externí:

```
<!DOCTYPE sestava [
<!ENTITY název
"Základní deska<tučně>ASUS A7N8X Deluxe</tučně>">
<!ENTITY seznam_čipů SYSTEM "čipseznam.xml">
<!ELEMENT obrázek_desky EMPTY>
<!ATTLIST obrázek_desky src ENTITY #REQUIRED>

<!NOTATION gif SYSTEM "ShowGIF.exe">
<!ENTITY asus SYSTEM "a7n8x.gif" NDATA gif>
]>
<sestava>
  <zákl_deska>
    &název;
  <obrázek_desky src="asus" />
    &seznam_čipů;
  </zákl_deska>
</sestava>
```

#### Parametrická interní-textová a externí-textová:

```
<!DOCTYPE sestava [
<!ENTITY seznam_elementu "(číslo, název, popis)">
```

```

<!ENTITY cdDTD SYSTEM "cd.dtd">
<!ENTITY knihaDTD SYSTEM "kniha.dtd">

<!ELEMENT CD %seznam_elementu;>
<!ELEMENT kniha %seznam_elementu;>
%cdDTD;
%knihaDTD;
]>

```

## CDATA sekce

Pokud potřebujeme vložit do XML dokumentu data, která obsahují mnoho zakázaných znaků, bylo by velice pracné nahrazovat je znakovými entitami, proto poskytuje specifikace XML nástroj, který tento problém řeší. Je to sekce CDATA, do které se vkládají data, která parser nebude interpretovat jako XML text.

```

<![CDATA[Instalace a konfigurace ovladače:
    <Krok 1>Instaluj ovladač do "/usr/drivers".
<Krok 2>Přejdi do konfiguračního souboru
    "/usr/drivers/nVidia/config.xml".
<Krok 3>Zruš komentář na řádce
    <!-- <config>/usr/drivers/nVidia/config.xml</config> -->
    -----> Použij Midnight commander <-----
]]>

```

### 3.10.3 XML jmenné prostory (namespaces)<sup>25</sup>

Jmenné prostory v XML fungují na bázi prefixů a jsou deklarovány jako hodnota atributu *xm:prefix* v elementu, který chceme zahrnout do daného prostoru. Tímto zahrneme do jmenného prostoru i všechny jeho potomky (podelementy). Hodnotou jmenného prostoru je unikátní URI<sup>26</sup> adresa. Jmenným prostorem (prefixem) můžeme označit i element, ve kterém je deklarace provedena. Pokud tedy chceme uplatnit nějaký jmenný prostor použijeme k tomu prefix, který musíme přiřadit ke každému elementu či atributu, který chceme do daného prostoru zahrnout. Můžeme určit i základní (defaultní), prostor jmen, který se uvozuje podobně, ovšem prefix se v tomto případě vynechá. Defaultní prostor jmen ale neplatí pro atributy, pouze pro všechny elementy

<sup>25</sup> [www.w3.org/TR/xml-names11/](http://www.w3.org/TR/xml-names11/)

<sup>26</sup> URI (Uniform Resource Identifier) je nejobecnější tvar identifikátoru nějakého datového zdroje. Existují i konkrétnější identifikátory a to URN (Uniform Resource Name) a URL (Uniform Resource Locator).

uvnitř elementu s deklarací. Jakýkoli jmenný prostor můžeme překrýt jiným jmenným prostorem.

```
<sestava xmlns="http://www.vlkcomputers.cz/sestavy"
  xmlns:dod="http://www.velkoobchod_comp.cz">
<dod2:zákl_deska xmlns:dod2="http://www.brnocomps.cz">
  <dod2:název>ASUS A7N8X Deluxe<název/>
  <dod2:čipset>nVidia NForce4</čipset>
</zákl_deska>

<dod:copyright>&vlkcomputers_Copyright;</dod:copyright>
</sestava>
```

## 3.11 Stanovení schéma XML dokumentu

Jelikož je XML rozšiřitelný jazyk a každý si může nadefinovat své vlastní libovolné značky, jeden a ten samý dokument se stejnými daty lze napsat mnoha různými způsoby. A v tom je právě zakopaný pes! Aplikace, které by měli zpracovávat takovýchle dokumenty by museli znát všechny možné značky, kterými je označen určitý údaj v těchto dokumentech. Napsat takovou aplikaci by jistě nebyla lehká záležitost.

Pokud tedy chtějí firmy nebo jiné organizace použít k vyměňování informací standard XML, musejí se dohodnout na přesné podobě svých dokumentů. K těmto účelům slouží dva standardy: *DTD* a *XML schéma*.

### 3.11.1 DTD

Jestliže XML dokáže plnohodnotně popsat data v XML dokumentu, DTD (XML Schéma) dělá z těchto dat použitelné vstupy (výstupy) pro různé typy aplikací. Pro začátek si ukážeme v přehledných bodech, co všechno DTD umí:

- Deklaruje sadu povolených elementů. Nelze pak v dokumentu použít jiné elementy, než ty, co jsou deklarovány v DTD. Ono je vlastně použít lze, ale potom už není dokument tzv. validní vůči tomuto DTD. Tento pojem si vysvětlíme později.
- Defínuje obsah každého elementu, pořadí elementů atd.

- DTD stanovuje sadu povolených atributů pro každý element, jejich jména, základní hodnoty atd.
- Poskytuje usnadnění tvorby modelu dokumentu pomocí entit nebo importu částí definice v podobě externího souboru.

### Ukázková DTD definice faktury [27]:

```
<!ELEMENT faktura (odběratel, dodavatel, položka+)>
<!ELEMENT dodavatel (název, adresa, ičo, dič)>
<!ELEMENT odběratel (název, adresa, ičo, dič)>
<!ELEMENT položka (popis?, cena, dph,cenaSDPH, ks?)>
<!ELEMENT název (#PCDATA)>
<!ELEMENT adresa (#PCDATA)>
<!ELEMENT ičo (#PCDATA)>
<!ELEMENT dič (#PCDATA)>
<!ELEMENT popis (#PCDATA)>
<!ELEMENT cena (#PCDATA)>
<!ELEMENT cenaSDPH (#PCDATA)>
<!ELEMENT dph (#PCDATA)>
<!ELEMENT ks (#PCDATA)>
<!ATTLIST faktura
    číslo CDATA #REQUIRED
    vystaveni CDATA #REQUIRED
    splatnost CDATA #REQUIRED
    vystavil CDATA #IMPLIED>
<!ATTLIST cena měna CDATA "CZK">
```

### Ukázka validního dokumentu (faktura.xml):

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet href="faktura.css" type="text/css"?>
<!DOCTYPE faktura SYSTEM "faktura.dtd">
<faktura číslo="F012" vystaveni="2.2.2000" splatnost="16.2.2000">
  <odběratel>
    <název>ZEOS s.r.o.</název>
    <adresa>Uhřická 21, Sedlec-Prčice 1, 257 91</adresa>
    <ičo>098744351</ičo>
    <dič>344-678549803</dič>
  </odběratel>
  <dodavatel>
    <název>Vlkcomputers, s.r.o.</název>
    <adresa>Svobody 263, Sedlec 1, 890 66</adresa>
    <ičo>6543890768</ičo>
    <dič>456-765476876</dič>
  </dodavatel>
  <položka>
```

```

    <popis>Vydláždění přístupové cesty k podniku</popis>
    <cena>50000</cena>
    <dph>19%</dph>
    <cenaSDPH>59500</cenaSDPH>
  </položka>
  <položka>
    <popis>Instalace Windows 2000 43 licencí</popis>
    <cena měna="USD">5000</cena>
    <dph>19%</dph>
    <cenaSDPH></cenaSDPH>
  </položka>
</faktura>

```

Definici typu dokumentu můžeme provést dvěma způsoby, které lze kombinovat. Buď definici umístíme uvnitř dokumentu nebo v externím souboru a nebo použijeme obě varianty. Externí umístění je výhodné z jednoho prostého důvodu. Pokud chceme validovat více XML dokumentů podle jednoho DTD nemusíme psát definici do každého zvlášť, ale stačí definovat typ dokumentu v jednom souboru s koncovkou *dtd* (u XML schémat *xsd*) a potom do každého dokumentu vložit odkaz na tento externí soubor. Deklaraci externího DTD jsme si ukázali při popisu hlavičky XML dokumentu v části „Stručný úvod do syntaxe jazyka XML“. Deklarace interního DTD se liší od externí v tom, že za kořenovým elementem je uvedena definice DTD v hranatých závorkách a ostatní údaje externí deklarace se vynechají.

```
<!DOCTYPE faktura[ ...definice DTD... ]>
```

Kombinace interní a externí definice se používá hlavně k doplnění nebo přepsání externího DTD, protože interní definice má přednost před externí. Příklad kombinace těchto dvou řešení vypadá takto:

```
<!DOCTYPE faktura SYSTEM "faktura.dtd" [ ...definice DTD... ]>
```

## Elementy

Definice elementů začíná otevírací DTD značkou „<!“ následovanou klíčovým slovem `ELEMENT` a názvem elementu. Za jménem elementu pak figuruje **model obsahu** elementu a vše doplňuje koncová značka „>“.

```
<!ELEMENT dodavatel (název, adresa, ičo, dič)>
```

Příklad definice elementu říká, že každý element  `Dodavatel`  musí obsahovat elementy  `název` ,  `adresa` ,  `ičo` ,  `dič`  právě jednou a v daném pořadí. Přitom tyto elementy by měli mít svoje vlastní definice umístěné dále v dokumentu. Element může obsahovat tyto prvky:

- **Další elementy** – elementy mohou obsahovat další zanořené elementy v určitém pořadí a výskytu,
- **Smišený obsah** – element v sobě uchovává jak textová data tak i elementy,
- **EMPTY** – při použití tohoto klíčového slova, element musí být prázdný,
- **ANY** – element může mít jakýkoli obsah.

V *modelu obsahu* lze použít různé symboly a kvantifikátory, pomocí kterých určíme například výskyt elementů nebo jejich pořadí. Kvantifikátory osvětluje následující tabulka.

**Tabulka 2** Kvantifikátory v DTD

KVANTIFIKÁTOR	POPIS
<b>Žádný kvantifikátor</b>	Jestliže za položkou není žádný kvantifikátor, element se může objevit právě jednou. (1)
*	Položka, za kterou se objeví tento kvantifikátor můžeme použít libovolněkrát. (0, ∞)
+	Jednou nebo vícekrát se může objevit položka, za kterou je tento znak. (1, ∞)
?	Tuto položku můžeme vynechat nebo použít nejvýše jednou. (0, 1)

Dále jsou k dispozici symboly „(“ „)“ „{“ „}“ „|““. Do závorek můžeme uzavřít několik položek, které lze následně kvantifikovat jako položku jednu. Pomocí čárky můžeme vyjádřit sekvenci položek (podobně jako operátorem AND) a znak svislé čáry vyjadřuje možnost výběru (podobně jako operátor OR).

```

<!DOCTYPE sestava [
<!ELEMENT faktura (odběratel, dodavatel, položka+)>
<!ELEMENT dodavatel (název, adresa, ičo, dič)>
<!ELEMENT odběratel (název, adresa, ičo, dič)>
<!ELEMENT položka ((číslo|čárkový kód),popis?, cena, dph, ks?)>
<!ELEMENT název (#PCDATA)>
<!ELEMENT adresa (#PCDATA)>
...
]>

```

Pokud element obsahuje pouze znaková data, uvádí se v modelu obsahu klíčové slovo #PCDATA (parsed character data). Tyto data procesor XML považuje za XML text, takže autor dokumentu se musí vyhýbat zakázaným znakům (&, <, ...).

```
<!ELEMENT dič (#PCDATA)>
```

### Smíšený obsah:

```

<!ELEMENT paměť (#PCDATA|velikost)*>
<!ELEMENT velikost (#PCDATA)>
<!ELEMENT paměť (#PCDATA)>

```

### Atributy

Definice atributů začíná otevírací DTD značkou „<!“ následovanou klíčovým slovem ATTLIST, názvem elementu, ve kterém má zamýšlený atribut figurovat a názvem atributu. Za jménem atributu se pak uvádí **typ** atributu a vše doplňuje koncová značka „>“.

**Tabulka 3** Stručný přehled klíčových slov pro typ atributu

KLÍČOVÉ SLOVO	POPIS
CDATA	Atribut obsahuje pouze znaková data.
ID	Atribut musí mít v každém elementu unikátní hodnotu.
IDREF, IDREFS	Hodnota tohoto atributu musí odpovídat hodnotě atributu (atributů oddělených mezerou v případě IDREFS) ID v jiném elementu, který je umístěn v dokumentu.
ENTITY, ENTITIES	Obsah atributu musí odpovídat názvu (názvům entit oddělených mezerou v případě ENTITIES) entity deklarované v DTD.
NMTOKEN, NMTOKENS	Hodnota atributu musí být určitý symbol, který může obsahovat písmena, číslice, tečky, pomlčky a podtržítka.

Pramen: J. Young, Michael – XML krok za krokem, 2002

```

<!ELEMENT faktura (odběratel, dodavatel, položka+)>
<!ATTLIST faktura
    číslo CDATA #REQUIRED
    vystaveni CDATA #REQUIRED
    splatnost CDATA #REQUIRED
    vystavil CDATA #IMPLIED>
<!ELEMENT položka (#PCDATA)>
<!ATTLIST položka
    idkód ID #REQUIRED
    spojenés IDREFS #IMPLIED>
<položka idkód="K454">Okap pozinkový</položka>
<položka idkód="K098">Okap měděný</položka>
<položka idkód="K658" spojenés="K454 K098">
Okap nerezový
</položka>

```

#### Vložení obrázku:

```

<!ELEMENT obrázek EMPTY>
<!ATTLIST obrázek src ENTITY #REQUIRED>
<obrázek src="UvodniObrazek" />

```

**Typ atributu můžeme určit i výčtem možných hodnot:**

```
<!ATTLIST sestava druh (notebook|desktop|PDA) "desktop">
```

Na místě hodnoty "desktop" v předešlém příkladě se mohou vyskytnout další klíčová slova, která opět vysvětluje tabulka.

**Tabulka 4** Vlastnosti a pravidla výskytu atributů

<b>KLÍČOVÉ SLOVO NEBO HODNOTA</b>	<b>POPIS</b>
#REQUIRED	Hodnota atributu musí být uvedena.
#IMPLIED	Atribut může být vynechán a XML procesor žádnou výchozí hodnotu nedoplní.
"výchozí hodnota"	Pokud je atribut vynechán, XML procesor doplní tuto výchozí hodnotu.
#FIXED "hodnota"	V případě vynechání atributu, se automaticky doplní hodnota v uvozovkách. Když je atribut uveden, nesmí mít jinou hodnotu, než tu, co je uvedena jako výchozí.

Pramen: J. Young, Michael – XML krok za krokem, 2002

**Validace dokumentu**

Co je to tedy ta validace? Když autor XML dokumentu potřebuje, aby dokument odpovídal určité struktuře, napíše DTD (XML schéma) a pomocí validace zjistí, zda odpovídá pravidlům uvedeným v tomto DTD (XML schématu). Pokud odpovídá, je dokument validní. Jakmile jsou ale pravidla DTD byla porušena, dokument už není **správně strukturovaný** (valid) podle tohoto DTD (XML schématu), ale přitom stále může být **syntakticky správný** (well-formed).

Tyto dva pojmy jsou velice důležité a znamenají, že pokud je dokument *validní*, je zároveň i *syntakticky správný*, ale v případě, že je jenom syntakticky správný, splňuje pouze syntaxi XML.

### 3.11.2 Stručný úvod do XML Schémat<sup>27</sup>

Vzhledem k tomu, že problematika XML schémat je velice rozsáhlá, pokusíme se v následující části pouze částečně nastínit možnosti XML schématu. XML schéma plní stejnou roli jako DTD. Omezuje nebo spíše vymezuje strukturu XML dokumentu. Oproti DTD má ale dvě hlavní výhody:

- datové typy jdou pomocí XML schématu určit mnohem konkrétněji a způsobů omezení struktury je neúměrně více,
- je to XML dokument, přesněji je to jedna z dalších speciálních aplikací XML a podléhá tedy syntaxi XML (pravidla vytváření XML schémat upravuje specifikace konsorcia W3C).

XML schéma se, narozdíl od DTD, nevkládá přímo do XML dokumentu, ale do samostatného souboru s koncovkou *xsd*. Kořenový element schématu musí mít jméno *schema* a musí patřit do jmenného prostoru *http://www.w3.org/2001/XMLSchema*.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="dodací_objednávka">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="vyřizovatel" type="xsd:string"/>
          <xsd:element name="komu">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="jméno" type="xsd:string"/>
                <xsd:element name="ulice" type="xsd:string"/>
                <xsd:element name="město" type="xsd:string"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="položka" maxOccurs="unbounded">
```

<sup>27</sup> [www.w3.org/TR/xmlschema-0/](http://www.w3.org/TR/xmlschema-0/)

```

<xsd:complexType>
<xsd:sequence>
  <xsd:element name="název" type="xsd:string"/>
  <xsd:element name="poznámka" type="xsd:string" minOccurs="0"/>
  <xsd:element name="množství" type="xsd:positiveInteger"/>
  <xsd:element name="cena" type="xsd:decimal"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
  </xsd:sequence>
  <xsd:attribute name="číslo_ob" type="xsd:string"
use="required"/>
  </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

### Ukázkový dokument validovaný podle uvedeného schématu:

```

<?xml version="1.0" encoding="ISO-8859-2"?>
<dodací_objednávka
číslo_ob="459923":xsi="http://www.w3.org/2001/XMLSchema-
instance":noNamespaceSchemaLocation="dodací_objednávka.xsd">
  <vyřizovatel>Jan Novák</vyřizovatel>
  <komu>
    <jméno>Jana Hlavínová</jméno>
    <ulice>Chelčického 25</ulice>
    <město>Praha 2</město>
  </komu>
  <položka>
    <název>C++ pro každého</název>
    <poznámka>Speciální edice</poznámka>
    <množství>1</množství>
    <cena>1000</cena>
  </položka>
  <položka>
    <název>PHP krok za krokem</název>
    <množství>1</množství>
    <cena>999</cena>
  </položka>
</dodací_objednávka>

```

Jak je patrné z příkladu, k odlišení hodnot patřících do jmenného prostoru jazyka XML schématu je použita předpona `xsd`. Tento prefix ale není povinný. Můžeme použít jakoukoli jinou vlastní předponu. Ukázkový dokument obsahuje informaci, která říká parseru, že dokument by měl být validován vůči schématu `dodací_objednávka.xsd`.

```

<dodací_objednávka
číslo_ob="459923":xsi="http://www.w3.org/2001/XMLSchema-
instance":noNamespaceSchemaLocation="dodací_objednávka.xsd">

```

## Deklarace elementů

Elementy jsou deklarovány pomocí elementu `xsd:element` a atributu `name`, který má hodnotu názvu elementu. *Kořenový element dokumentu* následuje ihned za *kořenovým elementem schématu*. Ostatní elementy jsou deklarovány uvnitř deklarace kořenového elementu dokumentu. Máme dva typy elementů: *jednoduchý* nebo *komplexní*. Jednoduchý typ může obsahovat pouze znaková data a komplexní typ může obsahovat navíc ještě další elementy (potomky) nebo atributy. Deklaraci těchto typů zapisujeme několika způsoby a my si v dalších odstavcích této kapitoly některé z nich ukážeme.

### Předdefinované a vlastní typy

Ve specifikaci XML schématu existuje několik *předdefinovaných typů*. Tyto typy můžeme použít například v deklaraci elementu jako hodnotu atributu `type` a určit tak typ dat, které element může obsahovat. Jednou z velkých výhod XML schémat je, že si lze nadefinovat i své *vlastní typy*. Technika definice vlastních typů spočívá v odvození z již předdefinovaných typů a v jejich omezení. Typ, ze kterého vlastní typ odvozujeme se označuje jako bazický a určíme ho atributem `base`. Následující tabulka popisuje některé předdefinované typy (kompletní přehled těchto typů je k dispozici na stránkách konsorcia W3C).

**Tabulka 5** Předdefinované typy určené specifikací XML schématu

TYP	POPIS
<code>string</code>	Obsahuje XML text.
<code>decimal</code>	Element obsahuje desetinné číslo (-6.3, 4, 7.0).
<code>integer</code>	Celé číslo (-6, 567, 0).
<code>positiveInteger</code>	Kladné celé číslo, kromě nuly (9, 324).

negativeInteger	Záporné celé číslo, kromě nuly (-987, -8).
time	Obsahuje časový údaj ve tvaru: hh:mm:ss.ss (18:34:43.00).
date	Datum ve tvaru rrrr-mm-dd (1955-12-30).
boolean	Pravdivostní hodnota, pravda a nepravda (True, False, 1, 0).
ID, IDREF, IDREFS	Stejné chování jako u DTD.

Pramen: J. Young, Michael – XML krok za krokem, 2002

### Příklad vlastního typu:

```
<xsd:element name="cena">
  <xsd:simpleType>
    <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="10"/>
      <xsd:maxExclusive value="unbounded"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

### Výskyt elementu

Výskyt elementu se udává pomocí atributů `minOccurs` a `maxOccurs`. Implicitní hodnota těchto dvou atributů je 1, tudíž při jejich vynechání to znamená, že každý element se může objevit právě jednou.

```
<xsd:element name="cena" type="xsd:decimal" minOccurs="0"
maxOccurs="2">
```

V příkladu vlastního typu se objevily i další elementy a atributy XML schématu, které si stručně popíšeme. V definici jednoduchého typu `simpleType` se objevily elementy `restriction`, `minExclusive` a `maxExclusive`, které mají za úkol omezit bazický typ `xsd:decimal` a vytvořit z něj typ, který my potřebujeme. Element

restriction se používá pro omezení bazických typů a elementy minExclusive a maxExclusive určují požadovaný interval z hodnot bazického typu xsd:decimal.

Další možností je atribut type vynechat a definovat typ uvnitř elementu. *Jednoduchý* typ je určený elementem simpleType a *komplexní* typ je určen elementem complexType.

```
<xsd:element name="cena">
  <xsd:simpleType>
    <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="10"/>
      <xsd:maxExclusive value="unbounded"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="datum">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="rok" type="xsd:integer"/>
      <xsd:element name="měsíc" type="xsd:integer"/>
      <xsd:element name="den" type="xsd:integer"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

#### Prázdný element:

```
<xsd:element name="konecřádku">
  <xsd:complexType>
</xsd:complexType>
</xsd:element>
```

## Deklarace atributů

Deklarace atributu se zapisuje pomocí elementu schématu xsd:attribute. Podobně jako u elementů, je tu možnost použití předdefinovaného typu nebo typu vlastního. Narozdíl od elementů, u deklarace vlastního typu lze použít pouze *jednoduchý typ* (simpleType).

```
<xsd:element name="adresa" type="údaje"/>
<xsd:complexType name="údaje">
  <xsd:sequence>
    <xsd:element name="jméno" type="xsd:string"/>
    <xsd:element name="bydliště" type="xsd:string"/>
    <xsd:element name="město" type="xsd:string"/>
    <xsd:element name="stát" type="xsd:string"/>
```

```

    </xsd:sequence>
    <xsd:attribute name="telefon" type="xsd:string"/>
</xsd:complexType>

```

### Deklarace vlastního typu atributu:

```

<xsd:attribute name="velikost">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="malá">
      <xsd:enumeration value="střední">
      <xsd:enumeration value="velká">
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>

```

### 3.11.3 Skladiště DTD a XML schémat (Repositories)

Jak jsme viděli v předešlých dvou částech této kapitoly, vytvoření kvalitního XML schématu či DTD je poměrně složitá a pracná záležitost, proto na Internetu vznikla jakási skladiště (repositories), která obsahují mnoho takovýchto XML slovníků. Navíc v oblasti jako je elektronická komerce (zejména B2B), jich přibývalo jako hub po dešti. Uvedme si dva příklady dnes již docela známých elektronických skladišť: **XML.ORG** a **BizTalk**.

## 3.12 Stylové jazyky

XML dokumenty ve své podstatě v sobě nemají žádné formátovací prvky a pokud programátor nebo webdesignér chce dokumenty, vyprodukované podnikovými nebo jinými aplikacemi, zobrazit nebo jinak prezentovat v aplikacích typu webového prohlížeče, PDA nebo mobilního telefonu či tiskárny, potřebuje k tomu nějaký nástroj, který řekne těmto aplikacím, jak mají elementy a atributy v XML dokumentu zobrazit. Nejznámějšími a asi i nejpoužívanějšími zobrazovacími a transformačními mechanismy jsou CSS a XSL. Problémem ale stále zůstává to, že různé aplikace podporují stylové jazyky odlišným způsobem nebo je nepodporují vůbec (např. podpora CSS ve webových prohlížečích).

### 3.12.1 Úvod do CSS<sup>28</sup>

CSS (Cascading Style Sheets) neboli kaskádové styly se používají již delší dobu. Byly totiž původně vyvinuty pro jazyk HTML, ale dnes se náramně hodí i pro formátování dokumentů XML. CSS styly se dají uložit do externího souboru (s koncovkou *css*), čímž splňují požadavky XML v oddělení obsahu od formátu dokumentu. XML dokument se na externí styl odkazuje prostřednictvím instrukce pro zpracování.

```
<?xml version="1.0" encoding="ISO-8859-2">
<?xml-stylesheet href="styl_sestavy.css" type="text/css"?>
<sestava>
...
</sestava>
```

Jako hodnota atributu `href` může být adresa URI, ale nejčastěji se používá adresa URL. Atribut `type` určuje MIME typ<sup>29</sup> použitého stylového jazyka. Pokud dokument potřebujeme zobrazit na různých médiích, přidáme do instrukce ještě atribut `media`. Tím řekneme procesoru, který ze stylů má na určité zařízení použít.

```
<?xml version="1.0" encoding="ISO-8859-2">
<?xml-stylesheet media="all" href="styl_sestavy.css"
type="text/css"?>
<?xml-stylesheet media="braille" href="styl_sestavy.css"
type="text/css"?>
<?xml-stylesheet media="screen" href="styl_sestavy.css"
type="text/css"?>
<?xml-stylesheet media="projector" href="styl_sestavy.css"
type="text/css"?>
<?xml-stylesheet media="tv" href="styl_sestavy.css"
type="text/css"?>

<sestava>
...
</sestava>
```

---

<sup>28</sup> Existuje již pracovní návrh specifikace CSS3, ovšem zatím se stále ještě pracuje na kandidátském doporučení verze CSS2.1, které je k dispozici na adrese <http://www.w3.org/TR/CSS21/>.

<sup>29</sup> Multipurpose Internet Mail Extensions (MIME) je internetový standard pro formát e-mailu. Prakticky všechen internetový e-mail je posílán prostřednictvím SMTP v MIME formátu.

Tabulka 6 Hodnoty atributu `media`

HODNTA ATRIBUTU	POPIS
<code>screen</code>	Obrazovka počítače.
<code>tty</code>	Obrazovka pracující pouze v textovém režimu.
<code>tv</code>	Obrazovka televizoru.
<code>projection</code>	Projektor.
<code>handheld</code>	Obrazovka kapesního zařízení.
<code>print</code>	Tiskárna.
<code>braille</code>	Hmatová čtečka Braillova písma.
<code>aural</code>	Hlasový syntetizátor.
<code>all</code>	Styl je vhodný pro všechna výstupní zařízení.

Pramen: Kosek, Jiří – XML pro každého podrobný průvodce, 1999

Styl se skládá z několika částí: *selektor* nebo seznam selektorů oddělených čárkou, složená závorka, *vlastnost*, dvojtečka, *hodnota vlastnosti* a uzavírací složená závorka. Vlastností v závorkách může být i více, oddělují se středníkem.

```
sestava zákl_deska, sestava paměť { text-align: center; ... }
```

Seznam selektorů říká, že vlastnosti uvedené v závorkách, platí pro všechny elementy `zákl_deska` a `paměť`, které jsou zanořeny v elementu `sestava`. Dále pak vlastnost `text-align` určuje zarovnání textu elementu. Hodnota `center` specifikuje zarovnání na střed.

Tvorba selektorů i seznam vlastností a jejich hodnot jsou podrobně rozvedeny ve specifikaci CSS [28].

```
položky {  
  font-family : Arial, Helvetica, sans-serif;  
  font-size : 13px;  
  display : table;  
}
```

```
nadpis {  
  display: block;  
  font-weight : bold;  
}
```

```
produkt {  
  display: block;  
}
```

```
název {  
  display : inline;  
}
```

```
cena {  
  display : inline;  
}
```

**Ukázkový dokument:**

```
<?xml version="1.0"?>  
<?xml-stylesheet href="styl.css" type="text/css"?>  
<položky>  
  <nadpis>Seznam položek</nadpis>  
  <produkt>  
    <název>Základní deska ASUS A7N8X Deluxe</název>  
    <cena>4678</cena>  
  </produkt>  
  <produkt>  
    <název>Grafická karta GForce FX5200</název>  
    <cena>2245</cena>  
  </produkt>  
  <produkt>  
    <název>Paměť OEM</název>  
    <cena>2500</cena>  
  </produkt>  
</položky>
```

### 3.12.2 Úvod do stylového jazyka XSL<sup>30</sup>

Stejně jako XML schémata i jazyk XSL je založen na syntaxi XML, což je opět nespornou výhodou. V přehledu „Příbuzné technologie XML“ jsem uvedl tři základní části specifikace XSL, které nyní podrobněji popíši.

#### XSLT (XSL Transformations)<sup>31</sup>

XSLT, jak říká specifikace, je jazyk určený pro transformaci XML dokumentů opět do XML dokumentů (jiné struktury). Samotnou transformaci provádí XSLT procesor. Krom toho může být použit i k transformacím do HTML nebo do dokumentu složeného právě z formátovacích objektů, které si popíšeme později. XSLT styl obsahuje elementy příkazy, které se používají pro sestavování cílového dokumentu a pro řízení XSLT procesoru. Jsou z jmenného prostoru *http://www.w3.org/1999/XSL/Transform*, podle kterého XSLT procesor pozná, že je má interpretovat. K označení těchto elementů se obvykle používá prefix `xsl`, ale není to pravidlem. Kořenovým elementem je `xsl:stylesheet`, do něj uvedeme deklaraci jmenného prostoru a volitelně i verzi stylu.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
...
</xsl:stylesheet>
```

#### Šablony

Základem stylů jsou šablony. Procesor prochází celý strom elementů (uzlů) vstupního dokumentu a pro každý element hledá šablonu ke zpracování. Šablonu vyjádříme elementem XSLT `xsl:template`. V šabloně je uvedeno na jakou část vstupního dokumentu bude aplikována a jak bude vypadat tato část ve výstupním dokumentu. Požadovanou úsek dokumentu vybereme pomocí atributu `match`. Tento atribut obsahuje výraz lokačního jazyka *XPath*. Tento jazyk bude stručně popsán v následující části. Pro jeho časté užití se předběžně zmíním o XPath výrazu „/“, který označuje kořenový element dokumentu.

<sup>30</sup> [www.w3.org/Style/XSL/](http://www.w3.org/Style/XSL/)

<sup>31</sup> [www.w3.org/TR/xslt](http://www.w3.org/TR/xslt)

```
<xsl:template match="/">
    <xsl:apply-templates/>
</xsl:template>
```

### Některé důležité elementy XSLT

V dokumentu lze použít dva druhy elementů – *řídící prvky* pro procesor a *elementy výsledného dokumentu* (např. HTML tagy).

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html>
    <body>
        <h2>Seznam položek</h2>
        <xsl:apply-templates/>
    </body>
    </html>
</xsl:template>
<xsl:template match="položky">
    <xsl:for-each select=".">
        <xsl:value-of select="produkt/název"/>
        <xsl:value-of select="produkt/cena"/>
    </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Přehled všech řídicích prvků (elementů) XSLT procesoru i jejich použití je k dispozici ve specifikaci XSLT na stránkách konsorcia W3C. My se nyní podíváme na ty nejpoužívanější.

#### **xsl:value-of**

Tento element se používá k přidání obsahu XML elementu do výstupního dokumentu. K vybrání elementu, jehož obsah se má použít se opět použije jazyk *XPath* a to v atributu `select`.

```
<xsl:template match="/">
    <xsl:value-of select="položky/název"/>
    ...
</xsl:template>
```

**xsl:for-each**

Tento XSLT element slouží k vybrání několika XML elementů podle speciální sady elementů uvedené v atributu `select`.

```
<xsl:for-each select="položky">
  <xsl:value-of select="produkt/název"/>
  <xsl:value-of select="produkt/cena"/>
</xsl:for-each>
```

**xsl:sort**

Element `sort` se používá k řazení elementů ve výstupním dokumentu. Klíč k řazení se uvádí v atributu `select` (opět XPath). Specifikace uvádí další atributy, které výběr a řazení upřesňují.

```
<xsl:for-each select="položky">
  <xsl:sort select="produkt/název"/>
  <xsl:value-of select="produkt/název"/>
  <xsl:value-of select="produkt/cena"/>
</xsl:for-each>
```

**xsl:if**

Tento element obsahuje šablony, které budou aplikovány, pouze v případě, že je splněna specifická podmínka v atributu `test` tohoto elementu. Atribut `test` musí obsahovat výraz, který má buď hodnotu `true` (pravda) nebo `false` (nepravda). Pravidla pro tvorbu onoho výrazu obsahuje specifikace *XPath 2.0*.

```
<xsl:for-each select="položky">
  <xsl:if test="produkt/cena > 5000">
    <xsl:value-of select="produkt/název"/>
    <xsl:value-of select="produkt/cena"/>
  </xsl:if>
</xsl:for-each>
```

**xsl:choose**

Tento element simuluje programovou konstrukci příkazu `switch` v běžných programovacích jazycích. Ovšem na místo klíčových slov `case` a `default` máme v XSLT elementy `xsl:when` a `xsl:otherwise`, které plní stejnou funkci. Element `xsl:otherwise` je nepovinný a element `xsl:when` se musí objevit alespoň jednou.

Podmínky splnění elementů `when` jsou v jejich atributu `test`. Tyto podmínky jsou postupně procházeny a v případě prvního splnění, element `choose` obsahuje pouze hodnotu tohoto `when` elementu. Pokud není splněna ani jedna podmínka v elementech `when`, je použit obsah elementu `otherwise`.

```
<xsl:for-each select="položky">
  <xsl:value-of select="produkt/název"/>
  <xsl:choose>
    <xsl:when test="cena > 10000">
      <xsl:value-of select="produkt/cena"/>
    </xsl:when>
    <xsl:when test="cena > 4000">
      <xsl:value-of select="produkt/cena"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="produkt/cena"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>
```

### **xsl:apply-templates**

Element `apply-templates` říká procesoru, že na aktuální element nebo jeho potomky se mají aplikovat šablony definované pro tento element. Když procesor žádnou šablonu nenajde, použije vlastní šablonu, která zkopíruje obsah elementu do výstupního dokumentu a spustí zpracování vnořených elementů.

```
<xsl:template match="položky">
  <xsl:apply-templates select="produkt/název"/>
  <xsl:apply-templates select="produkt/cena"/>
</xsl:template>

<xsl:template match="produkt/název">
  ...
</xsl:template>
<xsl:template match="produkt/cena">
  ...
</xsl:template>
```

### **xsl:element**

Do výstupního XML dokumentu lze přidávat elementy pouhým vložením do transformační šablony v XSL stylu. Pokud bude mít element další potomky, musí být

deklarovány uvnitř tohoto elementu. Jméno elementu udává hodnota atributu `name`. Můžeme také definovat jmenný prostor pomocí dalšího atributu `namespace`.

```
<xsl:element name="dph">
  <xsl:apply-templates/>
</xsl:element>
```

### **xsl:attribute**

Element `attribute` vytvoří nadřazenému elementu atribut. Název atributu určíme tradičně atributem `name` a jako u elementu můžeme použít i atribut `namespace` pro přiřazení k jmennému prostoru. Takto lze přidávat atributy nejen k elementům vloženým pomocí `xsl:element`, ale také k elementům zapsaným přímo.

```

  <xsl:attribute name="popis_obrazku">
    logo firmy
  </xsl:attribute>
</img>
```

### **XPath (XML Path Language)**<sup>32</sup>

Tento jazyk slouží k vybrání (lokaci) objektů v XML dokumentu. Pro zjednodušení přístupu k částem dokumentu považuje tento jazyk XML dokument za strom s uzly. Každému uzlu odpovídá buď element, atribut nebo obsah elementu. Vztahy mezi uzly fungují na bázi otce a předka. Následující tabulka ukazuje některé typy výrazů a jejich výběry.

**Tabulka 7** Výrazy jazyka XPath

VÝRAZ	VÝBĚR VÝRAZU
*	Všichni potomci aktuálního uzlu.
<code>zákl_deska</code>	Výběr se zúží na všechny elementy jménem <i>zákl_deska</i> , které jsou potomky aktuálního uzlu.

<sup>32</sup> [www.w3.org/TR/xpath](http://www.w3.org/TR/xpath)

@*	Lokalizuje všechny atributy aktuálního uzlu.
text ()	Výběr všech textových uzlů aktuálního uzlu.
@číslo	V aktuálním uzlu vybere atribut s názvem <i>číslo</i> .
/sestava/pev_disk[2]/název	Vzhledem k užití počátečního znaku „/“, výraz se vztahuje ke kořenovému uzlu a nikoli k aktuálnímu. Vybere <i>název</i> druhého pevného disku v sestavě.
.	Aktuální uzel.
zákl_deska[@serial="456ERFE"]	Vybere všechny elementy <i>zákl_deska</i> , které jsou děti aktuálního uzlu a mají atribut <i>serial</i> nastavený na „456ERFE“.
sestava[název="Kancelářská sestava"]	

Pramen: Kosek, Jiří – XML pro každého podrobný průvodce, 1999

V hranaté závorce mohou být různé funkce nebo operátory (relační, matematické), které dále výběr zúží.

### XSL-FO (XSL Formatting Objects)<sup>33</sup>

XSL-FO neboli XSL formátovací objekty představují jakýsi XML slovník, který používáme pro formát XML dat na obrazovku, papír, nebo jiné médium. Pro zjednodušení můžeme tento jazyk přirovnat k jazyku HTML a jeho formátovacím značkách, ovšem jazyk XSL-FO obsahuje mnohem více formátovacích nástrojů (značek) a je určen pro široký okruh médií.

<sup>33</sup> [www.w3.org/TR/2001/REC-xsl-20011015/](http://www.w3.org/TR/2001/REC-xsl-20011015/)

XSL-FO dokumenty jsou XML soubory s informacemi o vzhledu a obsahu požadovaného výstupu. Tyto soubory mají většinou koncovky *\*.fo* nebo *\*.fob*, ale protože to jsou také XML dokumenty, mohou mít také klasickou koncovku *\*.xml*.

XSL-FO dokument nemusíme vytvářet sami, můžeme použít styl XSL, který se aplikuje na zdrojový XML dokument. Výsledný dokument XSL-FO následně interpretujeme pomocí XSL FO procesoru do HTML, PDF nebo jiného souboru. Některé procesory umí rovnou z XML dokumentu pomocí XSL stylu vygenerovat výsledný soubor (PDF, HTML). Malá ukázka struktury XSL-FO dokumentu rozhodně neuškodí.

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

<fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
    <!-- Zde bude šablona stránky. -->
  </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="A4">
  <!-- Zde bude obsah stránky. -->
</fo:page-sequence>
</fo:root>
```

Sada značek XSL-FO jazyka musejí patřit do jmenného prostoru uvedeného v elementu `prefix:root` (většinou se používá prefix `fo`), kterým je obalen celý dokument s formátovacími objekty. Tento element má vždy alespoň dvě děti: `fo:layout-master-set` a `fo:page-sequence` elementy:

- **fo:layout-master-set** – tento element nám umožňuje určit základní vzhled stránek výsledného dokumentu. Je to například velikost stránky, zda použijeme záhlaví nebo zápatí stránky, jaké bude mít stránka okraje atd.,
- **fo:page-sequence** – v tomto elementu popisujeme co se má na jednotlivých stránkách objevit.

XSL procesor do definovaného prostoru stránky umísťuje obsah elementu `fo:flow`, do kterého se vkládají příslušné formátovací objekty (značky) popisující jednotlivé části výsledného dokumentu. Nejčastější z nich nám ukazuje následující tabulka.

Tabulka 8 Formátovací objekty

OBJEKTY	POPIS
float	Používá se pro plovoucí objekty, které jsou potřeba umístit na vhodné místo (obrázky, tabulky, ...).
block	Fo:block je určen k formátování odstavců, titulků, nadpisů, které mají blokový charakter.
inline	Tento element slouží k formátování částí, které se nemají chovat jako odstavce. Například při změně písma uvnitř odstavce.
simple-link	Poskytuje vkládání odkazů do výsledného dokumentu.
list-block, list-item, list-item-body, list-item-label	Tyto objekty se používají pro formát seznamů.
table, table-body, table-caption, table-cell, table-column, ...	Sada objektů, pro vytváření tabulek.
footnote	Poznámky pod čarou.

Pramen: Kosek, Jiří – XML pro každého podrobný průvodce, 1999

Každý formátovací objekt může mít několik atributů, kterými lze určit různé vlastnosti tohoto objektu. Mohou to být například: zarovnání, velikost písma, barva apod. Nabízí stejné vlastnosti jako kaskádové styly CSS a často mají i stejný název. Nyní si uvedeme ukázkou stylu pro generování FO.

```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format" version="1.0">
<xsl:template match="/">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master margin-bottom="2cm" margin-left="3cm"
margin-right="2cm" margin-top="2.5cm" master-name="my-
master">
        <fo:region-body/>
      </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-name="my-master">
      <fo:flow flow-name="xsl-region-body" font-family="Times Roman"
font-size="12pt">
        <fo:block>
          <xsl:apply-templates/>
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>
<xsl:template match="ident">
  <fo:block font-family="Helvetica" font-size="200%">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
<xsl:template match="název">
  <xsl:apply-templates/>
  <xsl:if test="following-sibling::podnázev">
    &#x2014;
  </xsl:if>
</xsl:template>
<xsl:template match="podnázev">
  <fo:inline font-style="italic">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>
<xsl:template match="šedě">
  <fo:inline color="gray">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>
<xsl:template match="zeleně">
  <fo:inline color="green">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>
</xsl:stylesheet>
```

### 3.13 Zpracování XML dokumentů

Technologie XML už není ve světě žádným nováčkem a rychle se šíří. Mnohé softwarové firmy a organizace zabývající se informačními technologiemi nelenily, a vyvinuly různé softwarové produkty pro tvorbu a zpracování XML dokumentů. Nemálo jich je volně šiřitelných a zdarma. Nejjednodušší je použít nejrůznější dostupná aplikační rozhraní (API) pro práci s XML. Uvedeme si ve stručnosti některé přístupy těchto rozhraní ke zpracování XML dokumentů.

#### 1) Sekvenční:

- rychlejší než stromové parsery, paměťově nenáročný,
- během průchodu XML dokumentem se nelze vracet,
- příklady rozhraní – SAX, pull-parsery.

#### 2) Stromový:

- pomalejší než sekvenční parsery, paměťově náročný (hlavně u velkých XML dokumentů),
- celý dokument je načten do paměti jako strom objektů,
- k dokumentu můžeme opakovaně, nesequenčně přistupovat,
- příklady rozhraní – DOM, JDOM.

Tyto aplikační rozhraní většinou mohou použít **různé druhy parserů**. Ty dokonalejší z nich umí dokument validovat vůči DTD či XML schématu, podporují jmenné prostory, XML katalogy nebo jsou platformově či jazykově nezávislí.

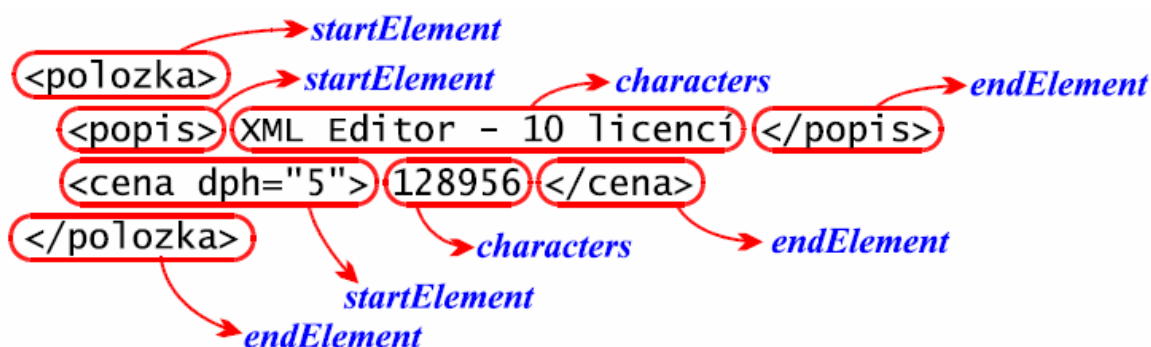
### 3.13.1 SAX<sup>34</sup>

SAX (Simple API for XML Parsing) není standardem konsorcia W3C, jak jsme si uvedli v podkapitole „Přehled příbuzných technologií XML“, ale klidně ho za standard považovat můžeme, protože se postupem času rychle rozšířil. Poslední verze (2.0.2) se liší od starší verze podporou jmenných prostorů.

Toto aplikační rozhraní je založeno na událostech. Během průchodu parseru dokumentem jsou generovány specifické události vztažené k určitému prvku (elementu, atributu, obsahu elementu, ...) XML dokumentu. SAX události:

- startDocument, endDocument
- startElement, endElement
- characters, processingInstructions
- startPrefixMapping, endPrefixMapping
- ignorableWhitespace, skippedEntity
- setDocumentLocator

Obrázek 2 SAX události



Pramen: <http://badame.vse.cz/izi238/prednasky.html>

<sup>34</sup> [www.saxproject.org/](http://www.saxproject.org/)

Podle toho, které části dokumentu potřebujeme zpracovat, obsloužíme jen ty události, které jsou těmito částmi (prvky) vyvolány. Obsluha událostí má podobu třídy implementující rozhraní `org.sax.ContentHandler`. Toto rozhraní nutí k implementaci všech metod k obsluze událostí. Většinou ale nepotřebujeme definovat všechny metody, a proto existuje třída `org.xml.sax.helpers.DefaultHandler`, která už má tyto metody definované jako prázdné a stačí je předefinovat. Uvedeme si příklad, který sice je v jazyce Java, avšak API je stejné i pro jiné programovací jazyky [24].

```
public class SaxApp extends DefaultHandler {
    public static void main(String[] args) {
        try {
            SAXParserFactory factory = SAXParserFactory.newInstance();
            factory.setValidating(false);
            SAXParser par = factory.newSAXParser();
            XMLReader rd = par.getXMLReader();
            SaxApp sapp = new SaxApp();
            rd.setContentHandler(sapp);
            rd.setErrorHandler(sapp);
            rd.parse("dokument.xml");
        }
        catch(Exception e) {
            System.err.println(e.getMessage());
        }
    }
    public void startDocument() throws SAXException {
        System.out.println("Zacatek dokumentu");
    }
    ...další implementované metody obsluhující události...
}
```

### 3.13.2 DOM<sup>35</sup>

DOM (Document object model) je jak už víme standart W3C, který si klade za cíl být nezávislý na programovacím jazyce. Definuje rozhraní pro manipulaci s dokumentem. Má několik verzí, kterým se říká levely.

1) DOM1 (level 1):

- DOM HTML – určen pro práci s HTML stránkou,
- DOM Core (jádro)– určen pro práci s obecným XML dokumentem.

2) DOM2 (level 2):

- DOM Traversal (průchod) and Range (řazení, sekvence) – podporuje práci s částmi dokumentu, průchod dokumentem,
- DOM Events (události) – umožňuje obsluhu událostí,
- Core, HTML – dále rozšířeny,
- DOM Style – práce s připojenými kaskádovými styly.

3) DOM3 (level 3):

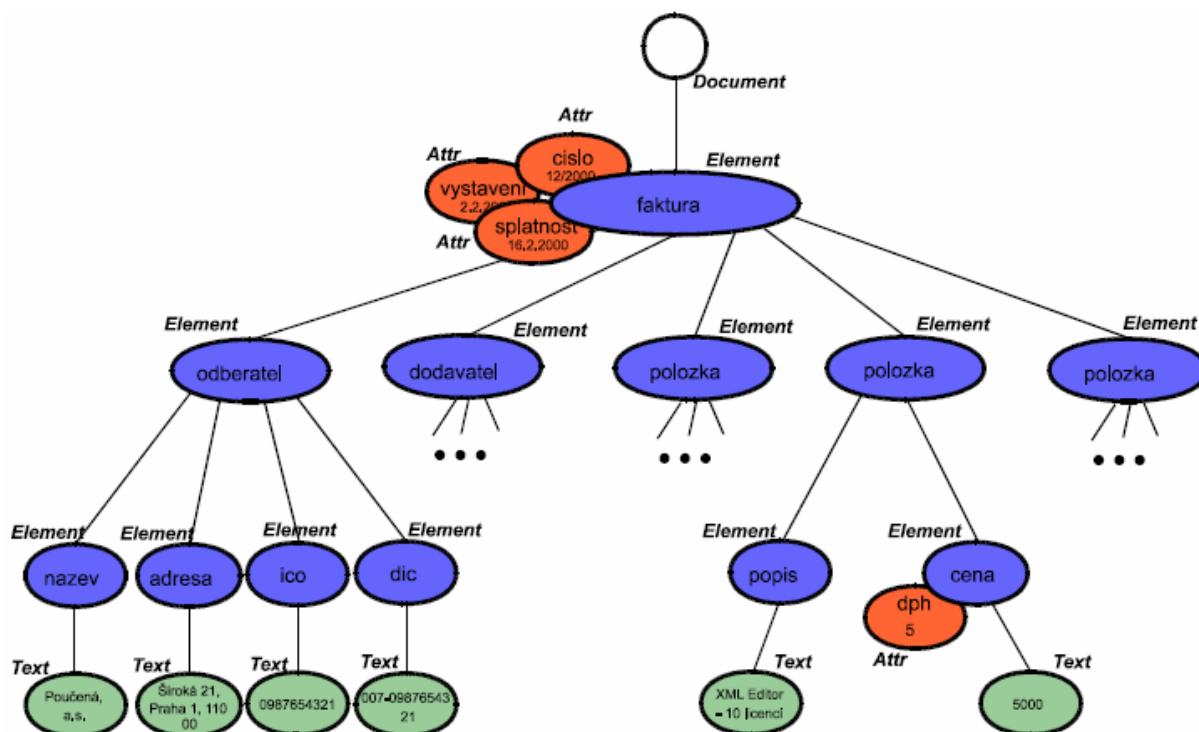
- DOM XPath – práce s XPath jazykem nad DOM stromem,
- DOM Load (načíst) and Save (uložit)– přídatné rozhraní na čtení a ukládání do souboru,
- DOM Validation (validace) – validace dokumentu (práce se schématem).

DOM rozhraní má mnohem vyšší paměťové nároky, než je tomu u rozhraní SAX. Je to způsobeno tím, že na začátku práce s dokumentem, načte celý dokument do paměti. Dokument v paměti má podobu stromu, který se skládá z elementů, atributů, jejich obsahu a dalších prvků dokumentu. Uzly stromu jsou reprezentovány objekty, se kterými můžeme manipulovat pomocí metod daných specifikací DOM.

---

<sup>35</sup> [www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407/](http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407/)

Obrázek 3 Strom vytvořený z dokumentu



Pramen: <http://badame.vse.cz/izi238/prednasky.html>

#### Některé častěji používané metody:

- **vytvoření uzlů** – `createElement()`, `createTextNode()`, `createComment()`,
- **přetváření stromu** – `insertBefore()`, `replaceChild()`, `removeChild()`, `appendChild()`,
- **procházení stromem** – `getChildNodes()`, `getParentNode()`, `hasChildNodes()`,
- **uzel** – `getNodeName()`, `getNodeValue()`, `getNodeType()`.

Následující příklad nám zobrazuje jak lze (opět v jazyce Java) vytvořit strom v paměti. Z objektu `DocumentBuilderFactory` vytvoříme objekt `DocumentBuilder`, který strom vytvoří. Objekt `Document` představuje kořen stromu [24].

```
public class DomApp {
    public DomApp() {
```

```

File fl = new File("dokument.xml");
try {
    DocumentBuilderFactory dbf =
        DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document dok = db.parse(file);

    Element koren = dok.getDocumentElement();
    ...zpracování dokumentu pomocí výše zmíněných metod uzlů...
}
catch(Exception e) {
    System.out.println("Chyba "+e.getMessage());
}
}
...
}

```

### 3.13.3 Stručný úvod do protokolu SOAP<sup>36</sup>

SOAP je jednoduchý protokol pro výměnu informací na bázi XML zpráv prostřednictvím síťového protokolu HTTP. Nahrazuje komunikaci používající vzdáleného volání procedur (RPC), tedy model požadavek/odpověď. Model RPC přináší problémy s kompatibilitou, bezpečností a firewally či proxy servery standardně blokují tento druh komunikace. Naproti tomu SOAP poskytuje platformovou a jazykovou nezávislost (na programovacím jazyce a technologii) a jde lepší cestou použití HTTP protokolu, který je podporován všemi internetovými prohlížeči a servery. Další výhodou SOAPu je, že je to W3C standard. SOAP zpráva je jednoduchý XML dokument obsahující následující elementy:

- **Envelope** (obálka) – je povinný kořenový element, identifikuje XML dokument jako SOAP zprávu,
- **Header** (hlavička) – volitelný element, který obsahuje hlavičkové informace,

<sup>36</sup> [www.w3.org/TR/soap/](http://www.w3.org/TR/soap/)

- **Body** (tělo) – povinný element, do kterého se vkládá požadavek a odpověď,
- **Fault** (chybný) – je element volitelný, který poskytuje informace o chybách, které nastaly během provádění zprávy.

#### Důležitá syntaktická pravidla tvorby SOAP zpráv:

- Musí být uložena v XML dokumentu,
- Musí používat jmenný prostor SOAP Envelope,
- Musí používat jmenný prostor SOAP Encoding,
- Nesmí obsahovat referenci na DTD,
- Nesmí obsahovat instrukce pro zpracování (processing instruction).

#### Kostra SOAP zprávy [21]:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

V předchozím příkladu vidíme, jaké jmenné prostory se používají pro označení elementů SOAP zprávy a jejich kódování. Oba jsou pro SOAP zprávu povinné.

Pro zjednodušení a zestručnění popisu SOAP zprávy, se budeme věnovat nejdůležitější části a tou je obsah elementu `soap:body`. Uvedeme si další příklad.

**SOAP požadavek:**

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:cena="http://www.burza.cz/cena">
    <cena:CenaAkcie>
      <cena:BurzovniJmeno>ZEOS</cena:BurzovniJmeno>
    </cena:CenaAkcie>
  </soap:Body>

</soap:Envelope>
```

**SOAP odpověď:**

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:cena="http://www.burza.cz/cena">
    <cena:CenaAkcieOdpoved>
      <cena:Cena>34.5</cena:Cena>
    </cena:CenaAkcieOdpoved>
  </soap:Body>

</soap:Envelope>
```

Příklady ukazují velmi jednoduchý SOAP požadavek na zjištění ceny akcie na burze s kódem ZEOS. Jednoduchá zpráva neobsahuje hlavičku, pouze tělo. V něm je požadavek na vyvolání vzdálené funkce `CenaAkcie` s parametrem pojmenovaným `BurzovniJmeno` s hodnotou ZEOS (kód akcií firmy ZEOS). Ukázky obsahují i HTTP hlavičky, které zde nebudeme dále zkoumat.

## 3.14 Praktická ukázka

Jednoduchá praktická ukázka se zaměřuje a elektronické obchodování typu B2C. Tato forma e-komerce je v České republice již poměrně častou záležitostí, ale většinou je založena na jiných technologiích než je XML (např. PHP, ASP, HTML, ...). Příložené CD k této práci obsahuje ukázku tohoto typu elektronického obchodování, ovšem s použitím XML. Jedná se o technologii Apache Cocoon [41], která si klade za cíl vytvořit jednoduché webové aplikační rozhraní založené na modulech, ze kterých se každá aplikace skládá.

## 4 Standardy elektronické komerce

V úvodu si připomeneme, že výhody standardizace e-komerce se využívaly již před uvedením jazyka XML a to v podobě EDI. S příchodem XML se implementace systémů EDI stala levnější, rychlejší a mnohem jednodušší. Pro urychlení a zjednodušení implementace EDI v elektronické komerci navíc vzniklo několik standardů na bázi XML, které mnoho implementačních činností zautomatizovalo a urychlilo.

### 4.1 Vybrané standardy e-commerce

Standardů zabývajících se elektronickou komercí je celá řada a opravdu nelze v této práci obsáhnout všechny. Za zmínku stojí třeba RosettaNet<sup>37</sup>, OAGIS<sup>38</sup>, ebXML, xCBL<sup>39</sup>, cXML nebo UBL. My si uvedeme příklady tří z nich, jeden je založený na DTD (cXML), jeden na XML schématech (UBL) a asi nejnámější standard **ebXML** prozkoumáme podrobněji.

#### 4.1.1 cXML (Commerce XML)<sup>40</sup>

Commerce XML je protokol založený na výměně standardizovaných obchodních XML dokumentů mezi dodavatelskými a odběratelskými aplikacemi (B2B). Tento protokol nezajišťuje plnou šíři interakce mezi obchodujícími stranami, ale poskytuje základní rámec komunikace, který si obchodující strany mohou různě přetvářet a doplňovat. CXML poskytuje jakousi sadu (volně dostupnou na webu cXML) standardizovaných dokumentů a DTD schémat, které jsou podobné tradičním papírovým dokumentům používaným při obchodování. Nejpoužívanější cXML dokumenty jsou:

- **Katalogy** – dokumenty, které předvádějí výrobky a služby odběratelským nebo nakupujícím organizacím. Poskytují ceny a popis zboží a služeb nabízených

---

<sup>37</sup> [www.rosettanet.org](http://www.rosettanet.org)

<sup>38</sup> [www.openapplications.org](http://www.openapplications.org)

<sup>39</sup> [www.xcbl.org](http://www.xcbl.org)

<sup>40</sup> [www.cxml.org](http://www.cxml.org)

dodavateli. Jsou hlavním komunikačním kanálem od dodavatelů k jejich zákazníkům,

- **PunchOut** – je pojem, který vyjadřuje komunikaci mezi aplikacemi, umožňující interakci s uživatelem na vzdáleném klientovi. Klasické e-komerční webové stránky podporující „PunchOut“ dokáží komunikovat s dodavatelskými systémy přes Internet a to prostřednictvím cXML,
- **Objednávky** – nakupující organizace zasílá objednávky jejím dodavatelům pro potvrzení obchodní transakce. K těmto účelům se používají i jiné formáty (např. ANSI X12 EDI 850), ale cXML je mnohem flexibilnější, levnější.

#### Ukázka dokumentu dle specifikace cXML (OrderResponse.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM
"http://xml.cXML.org/schemas/cXML/1.2.011/cXML.dtd">
<cXML payloadID="9949494@cxml.workchairs.com" xml:lang="en-US"
timestamp="1999-03-12T18:39:09-08:00">
  <Response>
    <Status code="200" text="OK"/>
  </Response>
</cXML>
```

Kompletní aktualizovanou specifikaci cXML nalezneme na webových stránkách cXML. Současná verze je k nahlédnutí na příloženém CD.

### 4.1.2 UBL(Universal Business Language)<sup>41</sup>

Jazyk UBL (Universal Business Language) je průmyslový standard vyvíjený organizací OASIS, který definuje standardizované obchodní XML dokumenty (objednávky, faktury) příslušnými XML schématy. UBL představuje jakýsi odrazový můstek do elektronické komerce pro malé a střední podniky.

Zatím existuje jenom jediná verze tohoto jazyka a to UBL 1.0, která byla uvedena na světlo světa 8. listopadu 2004. Specifikace je k dispozici zdarma na stránkách

<sup>41</sup> [www.oasis-open.org/committees/ubl](http://www.oasis-open.org/committees/ubl)

organizace OASIS. Současná verze je opět umístěna na příloženém CD. Archiv se specifikací UBL 1.0 mimo jiné obsahuje:

- **XML schémata pro osm základních obchodních dokumentů** – objednávka, odpověď na objednávku, zjednodušená odpověď na objednávku, změna objednávky, zrušení objednávky, avízo o odeslání, oznámení o příjmu, faktura,
- **popis obecného procesu od objednávky k faktuře** – včetně typů dokumentů UBL určených pro různé fáze tohoto procesu,
- **knihovnu více než 400 univerzálních XML elementů** – ze kterých jsou UBL schémata sestavena,
- **příklady dokumentů odpovídajících UBL schématům,**
- **detailní popis metod vývoje schémat kompatibilních s UBL** – v případech kdy si firma nebo společnost potřebuje upravit UBL schémata podle svých specifických potřeb.

### Ukázka dokumentu dle UBL (UBL-OrderCancellation-1.0-Office-Example.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<OrderCancellation
xmlns:stat="urn:oasis:names:specification:ubl:schema:xsd:DocumentS
tatusCode-1.0"
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasi
cComponents-1.0"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggr
egateComponents-1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="urn:oasis:names:specification:ubl:schema:xsd:OrderCancellat
ion-1.0"
xsi:schemaLocation="urn:oasis:names:specification:ubl:schema:xsd:O
rderCancellation-1.0 ../../xsd/maindoc/UBL-OrderCancellation-
1.0.xsd">

<ID>20031654-X</ID>
<IssueDateTime>2003-03-09T09:30:47</IssueDateTime>
<CancellationNote>order replaced</CancellationNote>
<cac:OrderReference>
  <cac:BuyersID>20031654-1</cac:BuyersID>
  <cbc:IssueDate>2003-03-07</cbc:IssueDate>
</cac:OrderReference>
<cac:BuyerParty>
```

```

    <cac:Party>
      <cac:PartyName>
        <cbc:Name>Bills Microdevices</cbc:Name>
      </cac:PartyName>
    </cac:Party>
  </cac:BuyerParty>
  <cac:SellerParty>
    <cac:Party>
      <cac:PartyName>
        <cbc:Name>Joes Office Supply</cbc:Name>
      </cac:PartyName>
    </cac:Party>
    <cac:OrderContact>
      <cbc:Name>Betty Jo Beoloski</cbc:Name>
    </cac:OrderContact>
  </cac:SellerParty>
</OrderCancellation>

```

### 4.1.3 ebXML (electronic business XML)<sup>42</sup>

Iniciativa ebXML byla založena v listopadu 1999 v San José v Kalifornii organizacemi UN/CEFACT (United Nations Body for Trade Facilitation and Electronic Business) a OASIS (Organization for the Advancement of Structured Information Standards). Základním cílem zakladatelů ebXML bylo a stále je, vytvořit globální elektronický trh, kde obchodní subjekty rozdílné velikosti a geografického umístění, mohou hledat své obchodní partnery a provádět obchodní transakce s jakýmkoli jiným obchodním subjektem prostřednictvím technologie výměny dat přes Internet založené na XML.

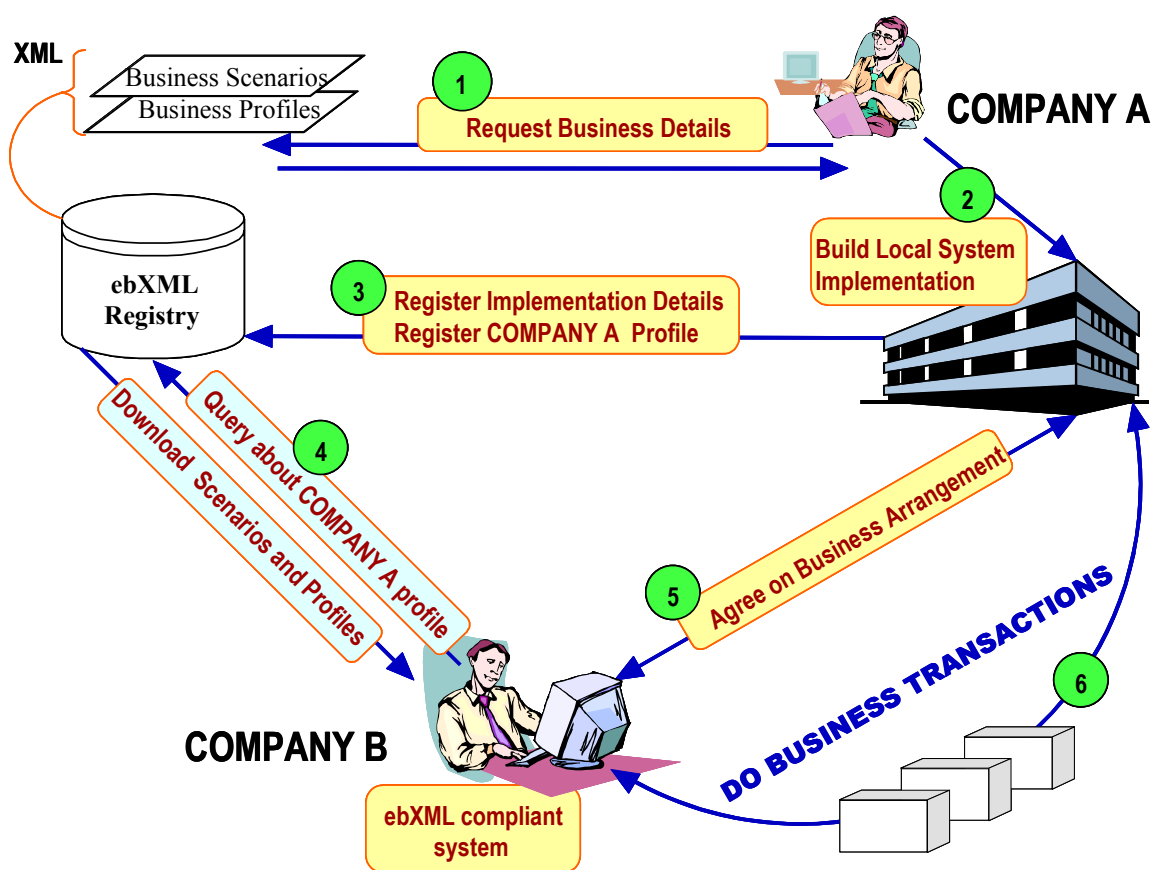
EbXML je sada specifikací tvořící celek souvisejících komponent, který automatizuje celý obchodní proces tj. od vyhledání obchodního partnera, přes dohodu o spolupráci, až po samotné provádění elektronických obchodních transakcí. Každá ze zakládajících organizací přispívá do ebXML svým dílem. UN/CEFACT se stará o stránku obchodní (tzv. core components – základní komponenty, modely obchodních procesů) a OASIS je zodpovědná za technickou stránku věci (posílání zpráv, registry/repozitory – registry/úložiště, implementace, technická spolupráce obchodních partnerů). Nyní si přiblížíme v základních bodech koncepci ebXML:

<sup>42</sup> [www.ebxml.org](http://www.ebxml.org)

- standardní mechanismus popisující *obchodní procesy* a informační modely těchto procesů,
- systém registrace a ukládání informací o *obchodních procesech, informačních meta modelech*, které jsou sdíleny a obnovovány,
- zpřístupnění informací (které *obchodní procesy* podporují, jaká *aplikační rozhraní* používají v obchodním procesu, typ používaných *zpráv, transportní vrstva, zabezpečení*) o každé registrované společnosti,
- mechanismus pro popis a provedení vzájemné obchodní dohody (Collaboration Protocol Agreement – CPA), která je často odvozována z informací o firmách (viz. předcházející bod),
- standardizovaný rámec *služby výměny informací* (Messaging Service) pro bezpečnou a spolehlivou výměnu zpráv mezi obchodními partnery,
- systém konfigurace služby výměny informací podle pravidel definovaných v obchodní dohodě.

Nejrychleji pochopíme jednotlivé ebXML fáze elektronického obchodování na jednoduchém příkladu scénáře spolupráce dvou firem.

Obrázek 4 Základní scénář spolupráce dvou firem



Pramen: Technická specifikace ebXML (verze 1.04)

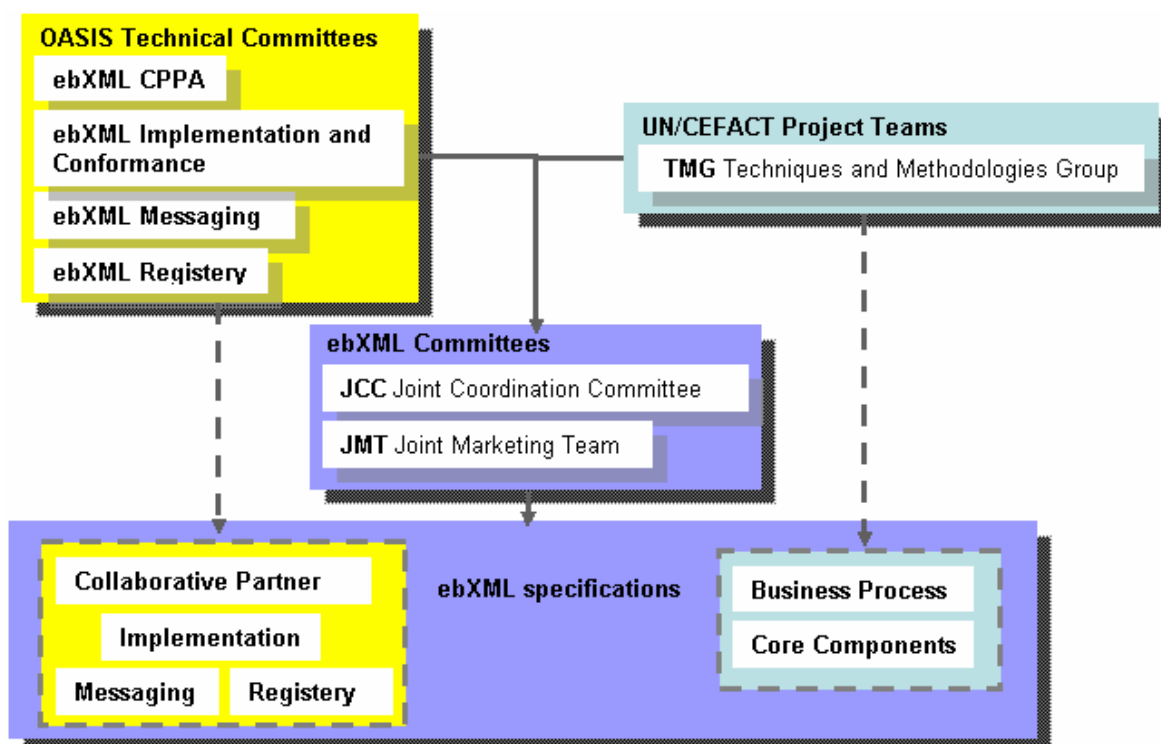
Firma A hledá nové obchodní příležitosti a prostřednictvím Internetu (nebo jiné sítě) si vyhledá v ebXML registrech popis požadovaných informací o svém oboru činnosti (krok 1). Na základě seznámení se s obsahem registru ebXML, se podnik A rozhodne pro vytvoření svojí vlastní aplikace, která je v souladu s ebXML (krok 2). Firma nemusí vyvíjet aplikaci sama, jsou k dispozici mnohá komerční řešení této implementace.

Firma A si následně zaregistruje svůj profil (Business Profile) v ebXML registrech (krok 3). Profil popisuje podporované obchodní scénáře a možnosti či omezení firmy v elektronickém obchodování. Tyto obchodní scénáře jsou XML verze obchodních procesů, které firma může provozovat. Pokud je formát a použití obchodních scénářů v souladu se standardem ebXML, firma A obdrží potvrzení o správnosti.

Firma B hledá vhodného obchodního partnera, prostřednictvím profilovaného dotazu na ebXML Registry (krok 4). Mimo jiné se jí dostane do rukou profil firmy A, který obsahuje veškeré potřebné informace. Obchodní scénáře firmy A jí plně vyhovují a rozhodne se, že chce s firmou A začít obchodovat. Zašle tedy firmě A sdělení (poptávku), že chce používat její obchodní scénáře používající ebXML (krok 5). Firma B implementuje svou ebXML odpovídající aplikaci. V rámci kroku 5 si také obě firmy vymění podrobnosti o obchodních scénářích, které budou používat. Poté, co firma A souhlasí se zahájením obchodu, jsou obě společnosti připraveny k elektronickému obchodování s využitím ebXML (krok 6).

Uvedené kroky v předchozím příkladu se provádějí pomocí částí (komponent) technologie ebXML. Na následujícím diagramu vidíme ze kterých komponent se ebXML skládá a z čeho vycházejí.

**Obrázek 5** Organizační schéma ebXML



Pramen: [www.ebxml.eu.orgCZ/ebxml/Iniciativy\\_ebXML.htm](http://www.ebxml.eu.orgCZ/ebxml/Iniciativy_ebXML.htm)

Kromě implementace (Implementation) jsou všechny části ebXML definovány příslušnými specifikacemi. V následujících odstavcích se s těmito komponentami ebXML technologie blíže seznámíme a na několika příkladech si ukážeme jejich funkci.

## Spolupracující partneři (Collaborative Partner)

Partneři v elektronickém obchodování by se měli nejprve dohodnout na podmínkách, za kterých budou provádět elektronické transakce. Specifikace (*Collaboration-Protocol Profile and Agreement Specification Version 2.0*), která se touto problematikou zabývá, stanovuje základní dokumenty, které by měli obě strany použít a co musí tyto dokumenty obsahovat. Jsou to: **Protokol dohody o spolupráci** – Collaboration-Protocol Agreement (CPA), a **Profil protokolu spolupráce** – Collaboration-Protocol Profile (CPP). CPA se většinou tvoří jako průnik dokumentů CPP spolupracujících stran, které firmy ukládají do registrů ebXML.

CPP definuje možnosti firmy v oblasti obchodní výměny zpráv. Naproti tomu CPA definuje společnou cestu interaktivní spolupráce, na které se zúčastněné strany dohodli, a kterou jsou oba partneři schopni dodržovat a vykonávat. Tento dokument by měli následně použít ke konfiguraci svých ebXML aplikačních systémů. Tato skutečnost jim zajistí kompatibilitu, při výměně zpráv, i když používají systémy od různých výrobců. Oba typy dokumentů jsou XML soubory, jsou tedy strojově čitelné. Kompletní dokumenty tohoto typu jsou velice rozsáhlé, proto si v následujících příkladech ukážeme pouze jejich kostry.

### Ukázka kostry dokumentu CPP:

```
<tp:CollaborationProtocolProfile
xmlns:tp="http://www.oasis-open.org/committees/ebxml-
cppa/schema/cpp-cpa-2_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-
cppa/schema/cpp-cpa-2_0.xsd http://www.oasis-
open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xlink="http://www.w3.org/1999/xlink"
tp:cppid="uri:companyA-cpp"
tp:version="2_0b">
<tp:PartyInfo> <!-- one or more -->
...

```

```

</tp:PartyInfo>
<tp:SimplePart id="..."> <!-- one or more -->
...
</tp:SimplePart>
<tp:Packaging id="..."> <!-- one or more -->
...
</tp:Packaging>
<tp:Signature> <!-- zero or one -->
...
</tp:Signature>
<tp:Comment>text</tp:Comment> <!-- zero or more -->
</tp:CollaborationProtocolProfile>

```

### Ukázka kostry dokumentu CPA:

```

<CollaborationProtocolAgreement
xmlns:tp="http://www.oasis-open.org/committees/ebxml-
cppa/schema/cpp-cpa-2_0.xsd"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xlink="http://www.w3.org/1999/xlink"
tp:cpaid="YoursAndMyCPA"
tp:version="2.0a">
<tp:Status tp:value="proposed"/>
<tp:Start>1988-04-07T18:39:09</Start>
<tp:End>1990-04-07T18:40:00</End>
<!-- ConversationConstraints MAY appear 0 or 1 time -->
<tp:ConversationConstraints tp:invocationLimit="100"
tp:concurrentConversations="4"/>
<tp:PartyInfo>
...
</tp:PartyInfo>
<tp:PartyInfo>
...
</tp:PartyInfo>
<tp:SimplePart tp:id="..."> <!-- one or more -->
...
</tp:SimplePart>
<tp:Packaging tp:id="..."> <!-- one or more -->
...
</tp:Packaging>
<tp:Signature> <!-- zero or one time -->
...
</tp:Signature>
<tp:Comment xml:lang="en-GB">any text</Comment><!-- zero or more -
->
</tp:CollaborationProtocolAgreement>

```

## Implementace

Jde o poskytování prostředků výrobcům softwaru na vývoj aplikací, které dodržují specifikace ebXML a jsou schopné spolupracovat.

## Servis pro předávání zpráv (Messaging)

Tuto část ebXML upravuje specifikace (*Message Service Specification Version 2.0*). Servis pro předávání zpráv (ebMS) definuje obal zpráv a schéma hlavičky dokumentu užívané k odesílání ebXML zpráv přes nějaký komunikační protokol (HTTP, SMTP, ...) a chování softwaru posílajícího a přijímajícího ebXML zprávy. EbMS je sada vrstvených rozšíření *SOAP zpráv* nebo *SOAP zpráv s přílohami*. Implementace této specifikace by mohla být např. plně nezávislá aplikace nebo integrovaná komponenta v nějakém rozsáhlejší obchodním systému. Zaručuje spolehlivý přenos zpráv bez závislosti na technologických řešeních.

## EbXML zpráva

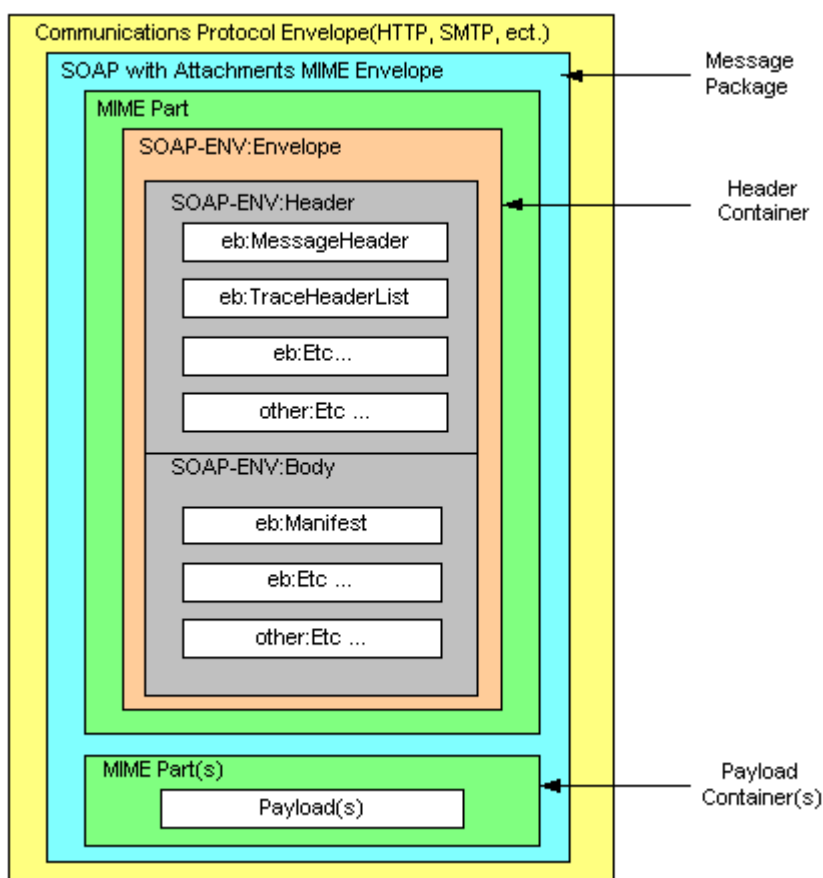
EbMS zavádí sadu rozšíření SOAP **hlavičky** (Header) a **těla** (Body) uvnitř SOAP **obálky** (SOAP Envelope). Zpráva se skládá z volitelného přenosového protokolu specifikovaného vnější komunikační obálkou (Communication Protocol Envelope) a schránky na zprávy (SOAP with Attachments MINE Envelope), jak můžeme vidět dále na obrázku „Obrázek 6“. Tato schránka používá MINE formát a je tvořena dvěma částmi:

- první MINE část „Header Container“ – obsahuje SOAP (SOAP 1.1) zprávu,
- druhá MINE část „Payload Container(s)“ – je nepovinná (nemusí být jenom jedna) a obsahuje data, která nejsou nijak limitována specifikací, může to být např. jednoduchý text nebo komplexní objekt, obsah každé takovéto další MINE části musí být identifikován v elementu `Manifest` uvnitř elementu `Body` SOAP zprávy.

Tyto části zprávy (Payload Containers) můžeme přirovnat k přílohám ve specifikaci SOAP s přílohami.

Hlavičkový kontejner (Header Container) tedy obaluje klasickou SOAP zprávu, ovšem uvnitř elementu `Header` figurují specifické ebXML hlavičkové elementy (`MessageHeader`, `TraceHeaderList`, ...) a uvnitř elementu `Body` jsou ebXML elementy s přenášenými daty nebo informacemi o dalších datech „Payloads“. Všechny uvedené skutečnosti jsou přehledně zobrazeny na obrázku.

**Obrázek 6** Struktura ebXML zprávy



Pramen: Message Service Specification Version 2.0

#### Ukázka ebXML zprávy založené na HTTP transportní vrstvě:

```
POST /servlet/ebXMLhandler HTTP/1.1
Host: www.example2.com
SOAPAction: "ebXML"
Content-type: multipart/related; boundary="Boundary"; type="text/xml";
start="<ebxmhheader111@example.com>"
--Boundary
Content-ID: <ebxmhheader111@example.com>
Content-Type: text/xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:eb="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-2_0.xsd"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
http://www.oasis-open.org/committees/ebxml-msg/schema/envelope.xsd
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
2_0.xsd
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
2_0.xsd">
<SOAP:Header>
<eb:MessageHeader SOAP:mustUnderstand="1" eb:version="2.0">
<eb:From>
<eb:PartyId>urn:duns:123456789</eb:PartyId>
</eb:From>
<eb:To>
<eb:PartyId>urn:duns:912345678</eb:PartyId>
</eb:To>
<eb:CPAId>20001209-133003-28572</eb:CPAId>
<eb:ConversationId>20001209-133003-28572</eb:ConversationId>
<eb:Service>urn:services:SupplierOrderProcessing</eb:Service>
<eb:Action>NewOrder</eb:Action>
<eb:MessageData>
<eb:MessageId>20001209-133003-28572@example.com</eb:MessageId>
<eb:Timestamp>2001-02-15T11:12:12</eb:Timestamp>
</eb:MessageData>
</eb:MessageHeader>
</SOAP:Header>
<SOAP:Body>
<eb:Manifest eb:version="2.0">
<eb:Reference xlink:href="cid:ebxmlpayload111@example.com"
xlink:role="XLinkRole" xlink:type="simple">
<eb:Description xml:lang="en-US">Purchase Order 1</eb:Description>
</eb:Reference>
</eb:Manifest>
</SOAP:Body>
</SOAP:Envelope>
--Boundary
Content-ID: <ebxmlpayload111@example.com>
Content-Type: text/xml
<?xml version="1.0" encoding="UTF-8"?>
<purchase_order>
<po_number>1</po_number>
<part_number>123</part_number>
<price currency="USD">500.00</price>
</purchase_order>
```

## **EbXML Registr (ebXML Registry/Repository)**

Registr je stabilní úložný mechanismus, kde jsou informace trvale uchovávány. Jsou to ty informace, které dopomáhají k uskutečnění B2B partnerství založené na ebXML. Mohou to být například XML schémata a dokumenty, popisy obchodních procesů, základní ebXML komponenty (ebXML Core Components), kontextové modely, UML modely, informace o firmách (CPP) a používaném softwaru.

Skutečná data těchto registrů se uchovávají v repositářích, které buď jsou a nebo nejsou fyzicky součástí registrů. EbXML registr slouží jako ukazatel na položky uložené v deponitářích. Tuto důležitou součást technologie ebXML upravují hned dvě specifikace:

- Specifikace služeb registru (*Registry Services Specification v2.1*) – definuje rozhraní služeb ebXML registru jako jsou komunikační protokoly, definice zpráv a XML schéma,
- Informační model registru (*Registry Information Model v2.1*) – definuje typy objektů uložených v registrech, jak jsou organizovány v registru; je založen na ebXML meta-modelech od různých pracovních skupin.

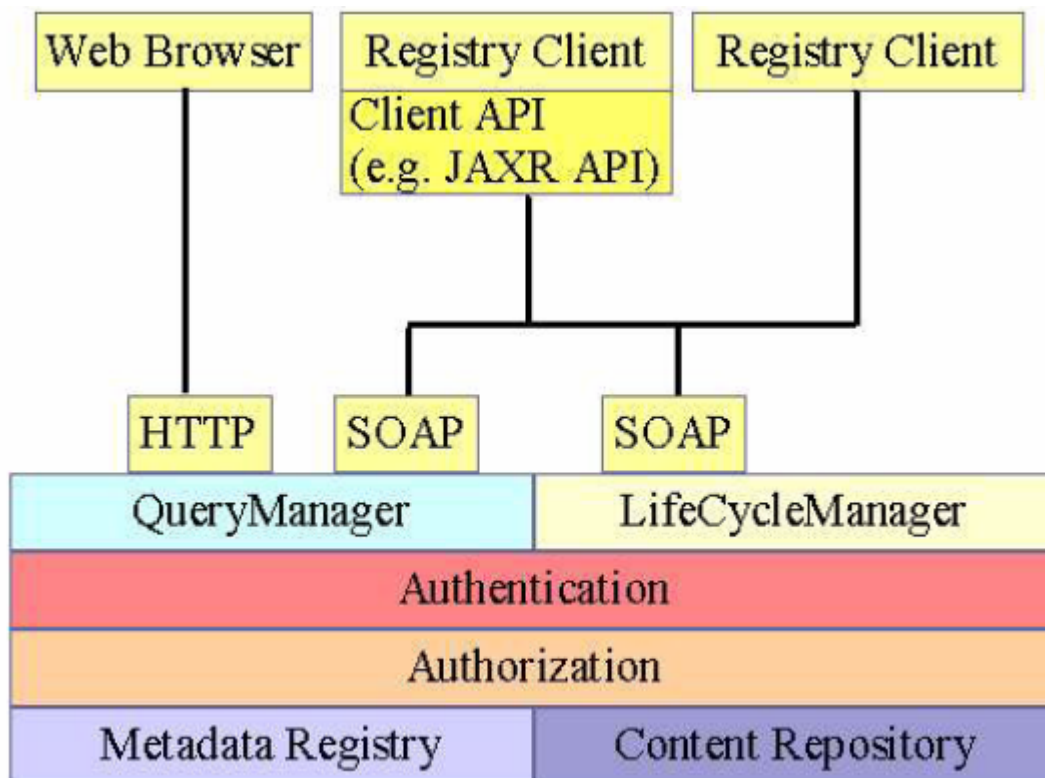
### **Specifikace služeb registru**

Tato specifikace je určena softwarovým vývojářům snažícím se o implementaci služeb ebXML registru a o vytvoření klienta ebXML registrů.

### **Informační model registru**

Tento model slouží opět hlavně implementátorům softwaru pro komunikaci s ebXML registrem. Mohou ho totiž využít v rozhodování o tom, jaké třídy (Classes) použít v jejich implementaci aplikace komunikující s ebXML registrem nebo jaký typ databáze je potřeba zahrnout do těchto aplikací. Následující obrázek ukazuje zjednodušený pohled na architekturu ebXML registrů a jejich fungování.

Obrázek 7 Zjednodušený pohled na architekturu ebXML registrů



Pramen: EbXML Registry Services and Protocols Committee Draft 01, 10. února, 2005

### Obchodní procesy (Business process)

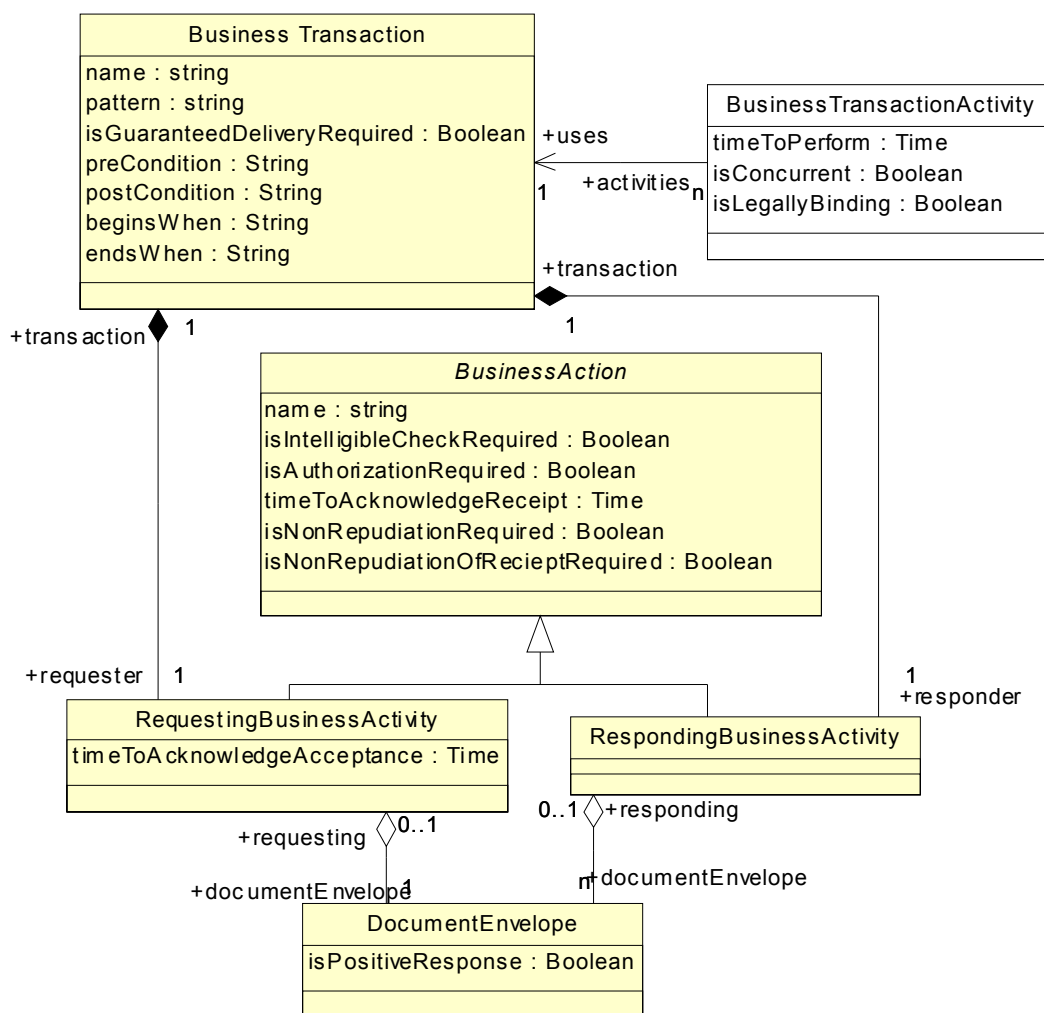
Obchodní činnosti se v různých odvětvích liší, proto bylo pro potřeby ebXML nutné zvolit nějaký mechanismus pro popis těchto činností. Pokud jsou totiž obchodní procesy korektně popsány, usnadní to automatizaci a převod těchto činností do elektronické podoby (konfiguraci softwarových komponent). Tyto modely používají pro definici aktivit a komunikace mezi podniky příslušné obchodní termíny.

K dispozici jsou modelovací jazyk UML a možnost převodu těchto činností do XML podoby. V případě UML verze se jedná o diagram tříd, které jsou propojeny a mají k sobě určitý vztah. U XML verze jde o DTD nebo W3C XML schémata ebXML obchodních procesů.

Obě verze (vyjadřující ten samý obchodní proces) jsou zaměnitelné a je jednoznačně určen vzájemný převod (konverze) z jedné verze do druhé.

Tyto modely samy o sobě neurčují strukturu obchodních dokumentů, ale jsou používány ve spojení s již existujícími definicemi obchodních dokumentů nebo s meta-modely obchodních dokumentů definovaných *ebXML základními komponentami* (ebXML Core Components).

**Obrázek 8** Ukázka UML modelu obchodní transakce



Pramen: Business Process Specification Schema verze 1.01

## Základní komponenty (Core Components)

Základní komponenty ebXML jsou datové objekty, které umožňují meziodvětvovou výměnu obchodních dokumentů. Mnoho slovníků, které se snaží standardizovat podobu určitých obchodních dokumentů, narážejí na nekompatibilitu v názvech stejných předmětů obchodní činnosti (výrobku nebo ceny). Různá odvětví, která si předávají své produkty je různě nazývají, i když se jedná o identický objekt. *Základní komponenty* se toto snaží řešit tím, že objektům používaným v obchodních procesech přidruží standardizovaná základní jména, ke kterým dále přiřazují další synonyma, a jedinečné identifikátory (UID – Universal Identifier). Tyto komponenty (standardizovaná podstatná jména) jsou uloženy v ebXML registrech v podobě slovníku základních komponent, na které se obchodní partneři (z různých odvětví) mohou odkazovat. Důležitou roli hraje identifikátor UID. Například obchodní partneři, kteří obchodují s určitým výrobkem, který nazývají různým způsobem, mohou použít základní komponentu (uloženou v ebXML registrech), která tomuto objektu (výrobku), přiřadí jednotný název a *jedinečný identifikátor* UID, na který se pak obě strany mohou odkazovat ve schématech svých XML dokumentů. Tím je zajištěno, že i když je výrobek nazýván v každém XML dokumentu jinak, jde stále o ten samý objekt.

### Ukázka ze specifikace Core Components Dictionary (verze 1.04):

<i>Category Type</i>	<b>Basic</b>	
<i>Core Component Type</i>	<b>amount. type</b>	
<i>Name</i>	charge price. amount	<i>definition</i>
<i>UID</i>	000127	The amount of money expected, required, or given in payment for something.
<i>UID</i>		
<i>Datatype</i>	n/a	
<i>Core Component Type</i>	amount. type	
<i>Core component re-used</i>	n/a	
<i>Synonyms</i>	price amount	<i>remarks</i>
<i>Naming Convention</i>		
<i>object class</i>	charge price	
<i>property term representation type</i>	amount*	
	amount	

<b>Name</b>	chargeable. amount	<b>definition</b>
<b>UID</b>	000145	The amount on which the charge is made.
<b>Datatype</b>	n/a	
<b>Core Component Type</b>	amount. type	
<b>Core component re-used</b>	n/a	
<b>Synonyms</b>		<b>remarks</b>
<b><u>Naming Convention</u></b>		
<b>object class</b>	chargeable	
<b>property term representation type</b>	amount*	
	amount	

Jak vidíme z předcházejícího přehledu specifikací, na kterých iniciativa ebXML staví, ebXML se neomezuje pouze na dohodu dvou obchodních partnerů o vyměňovaných dokumentech a jejich standardizovaném formátu. EbXML jde mnohem dál. Snaží se řešit širokou škálu problémů, které souvisejí s uzavíráním dohod o elektronickém obchodování a jeho provozem. Dělá černou práci, kterou podniky, nepoužívající ebXML, musejí dělat sami. Kompletní sada současných verzí ebXML specifikací je k této práci také přiložena na CD.

#### 4.1.4 Webové služby (Web Services)

V posledních několika letech se začala vyvíjet nová technologie, která může hodně promluvit (a již promlouvá) do světa elektronického obchodování, zejména do obchodování typu B2B. Jsou to tzv. *webové služby*. Co to jsou webové služby? Existuje mnoho definic, my si uvedeme jednu z nich, která je podle mého názoru nejužitečnější.

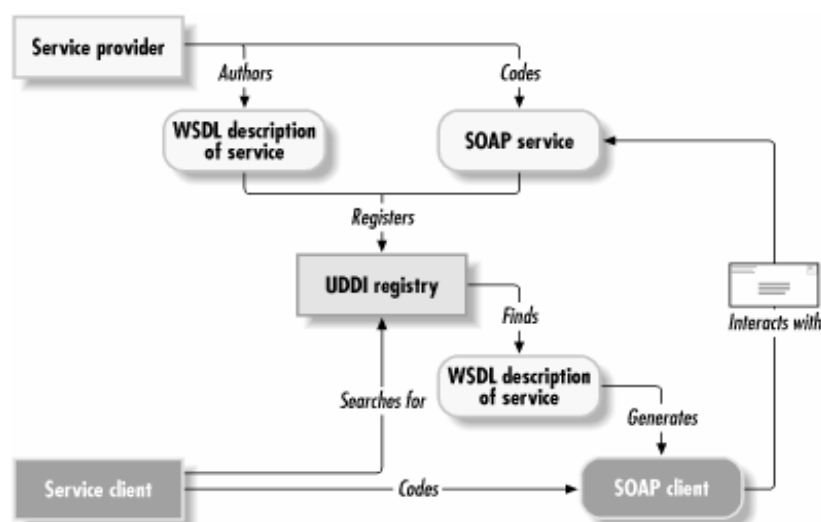
*„Webové služby (WS - web services) představují posun od velkých monolitních struktur aplikací k modelu založenému na komponentech. Aplikace jsou v rámci tohoto modelu sestavené z malých stavebních prvků - jednotlivých funkcí. Pokud jsou tyto funkce umístěné na různých internetových serverech, označují se jako webové služby. Takto sestavené aplikace je možné snadno vytvořit, dynamicky modifikovat a měnit. S použitím XML pro výměnu dat, dovolují webové služby spojit různé aplikace bez ohledu*

na počítačovou platformu, použité programovací jazyky a síťové protokoly a provozovat tak například e-business.“ [22].

Jedná se o jakési znovuzrození technologií pro vzdálené volání funkcí v distribuovaných systémech [23]. Tři základní technologie, na kterých webové služby budují svou funkcionalitu jsou:

- **SOAP** – protokol pro vzdálené volání procedur pomocí zpráv v XML (viz výše),
- **WSDL (Web Services Description Language)**<sup>43</sup> – jazyk popisující konkrétní webové služby; možnost vzdáleně volat funkce pomocí SOAP je k ničemu, pokud nevíme, jaké funkce se dají zavolat, jaké mají parametry a jaké vrací hodnoty; takovéto problémy řeší právě tento jazyk založený na XML,
- **UDDI (Universal Description, Discovery and Integration)**<sup>44</sup> – mechanismus pro nalezení požadovaných služeb, v únoru 2005 byl registr UDDI ratifikován jako standard organizace OASIS; nabízí veřejnou databázi, do které poskytovatelé webových služeb mohou ukládat popisy svých služeb a uživatelé je v ní mohou zase hledat.

Obrázek 9 Ukázka jednoduché interakce webových služeb



Pramen: McLaughlin, Brett – Java & XML Second Edition

<sup>43</sup> [www.w3.org/TR/wsdl20-primer/](http://www.w3.org/TR/wsdl20-primer/)

<sup>44</sup> [www.uddi.org/](http://www.uddi.org/)

Na obrázku „Obrázek 9“ je znázorněn základní princip webových služeb. Jedná se o komunikaci mezi *poskytovatelem služeb* (service provider) a *klientem služeb* (service client). Celý proces začne tím, že poskytovatel vytvoří (naprogramuje) svou webovou službu (serverový kód služby). Tuto službu popíše pomocí WSDL jazyka (ve WSDL dokumentu) a zaregistruje jí do UDDI registru (nebo do více takovýchto registrů), kde si jí mohou klienti, kteří mají o tuto službu zájem, vyhledat. Klient, který se rozhodne službu využívat, si vygeneruje (vytvoří) z WSDL popisu služby klientský kód, který umí tuto službu používat. V poslední fázi procesu webové služby se provádí interaktivní komunikace (volání služeb klientem) pomocí SOAP zpráv.

### **Elektronický obchod a webové služby**

Je zřejmé, že webové služby jsou postavené na vzájemné globální interakci aplikací. Toho lze jednoduše využít při komunikaci a **výměně obchodních informací** mezi aplikacemi obchodních partnerů (B2B). Takováto interakce povede, stejně jako u ebXML, k zjednodušení elektronického obchodování a ke snížení nákladů. Webové služby se dají použít i uvnitř podniku k podobným účelům. Vnitropodnikový tok informací by se opět zprůhlednil a zjednodušil.

Jedna z dalších možných uplatnění webových služeb by mohla být transformace dosavadních proprietárních vnitropodnikových nebo mezipodnikových systémů na webové služby. Pozbyly by tak platformové závislosti a otevřely by se systémům založeným na jiných platformách nebo programovacích jazycích.

### **4.2 EbXML a webové služby**

EbXML a webové služby jsou z pohledu elektronického obchodování komplementárními technologiemi. Zatímco webové služby se převážně hodí pro obchodně informační a integrační služby, nejsou konstruovány pro adekvátní podporu obchodně transakčních služeb, pro které se naopak hodí ebXML. Rozdíly v ebXML a webových službách ukazuje tabulka [20]:

**Tabulka :** Rozdíly mezi ebXML a webovými službami

<b>VLASTNOSTI</b>	<b>WEBOVÉ SLUŽBY</b>	<b>EBXML</b>
Typ	požadavek/odpověď	spolupráce
Komunikace	RPC (Remote procedure control) styl synchronní komunikace mezi těsně svázanými službami nebo v podobě asynchronního zasílání dokumentů mezi volně vázanými službami	synchronní a asynchronní komunikace
Popis rozhraní obchodní činnosti	WSDL	CPP, CPA
Protokol a formát	SOAP, XML	ebXML služba posílání zpráv (ebXML Message Service) prostřednictvím SOAP, XML, BPSS (jako „business“ protokol)
Standardy obsahu zpráv	žádné	doporučené standardy (OAGI BODs, ...)
Hledání obchodních partnerů	UDDI registr	ebXML registr (UDDI registr může odkazovat na ebXML registr)

Pramen: [www.webservices.org](http://www.webservices.org)

Technologie webových služeb má svou nejsilnější stránku v integraci, na umístění nezávislých, obchodních služeb typu požadavek/odpověď. Typickým příkladem takové služby je zjištění ceny akcií na burze.

Vypadá to, že iniciativa ebXML nebo webové služby mají svou budoucnost. V České republice se ale zatím vůbec nerozšířily. Hodně záleží také na kultuře obchodování v zemi a vůli k zavádění nových technologií. České firmy jsou v tomto ohledu zatím spíše opatrné a méně otevřené. Hodně působí také fakt, že tyto nové technologie nejsou zatím úplně propracovány (například v oblasti bezpečnosti) a otestovány.

## Závěr

Cílem této práce bylo analyzovat možnosti jazyka XML a jeho využití v elektronickém obchodování (ve výměně dat mezi aplikacemi elektronického obchodu). Dalším cílem bylo nalezení výhod jazyka XML oproti starší technologii EDI, která se v elektronickém obchodování používá již delší dobu.

V úvodních dvou kapitolách jsem se zabýval strukturou a podstatou elektronického obchodování a snažil jsem se přehledně a srozumitelně osvětlit podstatu činností v elektronické komerci a vyzdvihnout problémové oblasti.

Kromě analýzy jazyka XML jsem se pokusil o zmapování technologií, které se točí okolo XML a jsou důležitými pomocníky při implementaci XML nejen v e-komerci. Dále jsem uvedl asi nejnadějnější standardy elektronické komerce, které se pomalu začínají používat v praxi (cXML, UBL, ebXML a webové služby).

Technologie „XML v e-komerci“ ještě úplně nedozrála, ale již nyní lze s jistotou říci, že v blízké budoucnosti se rozšíří, jako jiné podobně revoluční technologie (HTML, PHP, ASP, .NET, ...).

Díky použití XML v e-komerci bude elektronické obchodování přístupné i malým a středním firmám a způsob obchodování bude pružnější, rychlejší a efektivnější.

## Slovník použitých zkratk

**ANSI** (American National Standards Institute) – Americký národní úřad pro normalizaci.

**API** (Application Programming Interface) – Programové aplikační rozhraní.

**ASP** (Active Server Pages) – Aktivní stránky serveru. Technologie firmy Microsoft, která umožňuje zpracovat Java a Visual Basic skripty uložené v HTML stránkách na serveru a klientovi odeslat výsledek.

**B2B** (Business-to-Business) – Obchodování mezi firmami. Zpravidla vyjadřuje mezi-podnikové vztahy, operace nebo transakce.

**B2C** (Business-to-Consumer) – Prodej koncovému zákazníkovi (spotřebiteli). Zpravidla vyjadřuje vztahy, operace nebo transakce mezi obchodem (firmou) a koncovým zákazníkem.

**C2B** (Consumer-to-Business) – Vztah koncového zákazníka (spotřebitele) k firmě. Zpravidla vyjadřuje vztahy, operace nebo transakce mezi koncovým zákazníkem a obchodem (firmou).

**C2C** (Consumer-to-Consumer) – Vztah mezi koncovými zákazníky (spotřebiteli). Burzy, výměny aukce apod., kde prostředníkem mezi nabízející a poptávající osobou je Internet.

**CRM** (Customer Relationship Management) – Správa vztahů se zákazníky. Komplexní správa a péče o zákazníky firmy.

**CSS** (Cascading Style Sheets) – Kaskádované styly. CSS jsou W3C standardem rozšiřujícím HTML standard. Umnožňují pokročilé formátování na WWW stránkách.

**cXML** (Commerce XML) – Protokol založený na výměně standardizovaných obchodních XML dokumentů mezi dodavatelskými a odběratelskými aplikacemi (B2B). Poskytuje sadu (volně dostupnou na webu cXML) standardizovaných dokumentů a DTD schémat, které jsou podobné tradičním papírovým dokumentům.

**DOM** (Document Object Model) – Model sestávající z jednoho kořene, elementů a jejich atributů.

**DSSSL** (Document Style Semantics and Specification Language) – Jazyk sémantiky a specifikace stylu dokumentu. Stylový jazyk původně vyvinutý pro potřeby jazyka SGML.

**DTD** (Document Type Definitions) – Definice typu dokumentů.

**ebXML** (Electronic Business XML) – XML pro elektronický obchod. Iniciativa z roku 1999 pro bezpečnou výměnu obchodních informací založená organizacemi OASIS UN/CEFACT.

**EDI** (Electronic Data Interchange) – Elektronická výměna dat. Systém výměny elektronických strukturovaných dokumentů mezi aplikacemi nezávislých objektů.

**EDMS** (Electronic Document Management Systems) – Systémy správy elektronických dokumentů.

**ERP** (Enterprise Resource Planning) – Plánování podnikových zdrojů. Podnikový informační systém pro koordinaci prodeje a objednávek s výrobou.

**HTML** (HyperText Markup Language) – Označovací jazyk pro hypertext. Jazyk používaný pro vytváření hyper-mediálních dokumentů pro službu WWW Internetu. Podmnožina SGML.

**HTTP** (HyperText Transfer Protocol) – Hypertextový přenosový protokol. Protokol používaný službou WWW síť Internet.

**ISO** (International Standards Organization) – Mezinárodní organizace pro normalizaci.

**JAXP** (Java API for XML Processing) – Jednoduché aplikační rozhraní pro XML v jazyce Java.

**JDOM** (Java Document Object Model) – Knihovna, která si klade za cíl, zjednodušit manipulaci s XML dokumenty v programovacím jazyce Java.

**MIME** (Multipurpose Internet Mail Extension) – je internetový standard pro formát e-mailu. Prakticky všechny internetový e-mail je posílán prostřednictvím SMTP v MIME formátu.

**MP3** (Music Protocol 3) – Hudební protokol 3. Datový formát pro digitální záznam zvuku vyznačující se vysokým stupněm ztrátové komprese při zachování zvukové kvality.

**OAGIS** (Open Applications Group, Inc. Standard) – XML standard pro B2B komerci, který lze použít v ebXML nebo webových službách.

**OASIS** (Organization for the Advancement of Structured Information Standards) – Nezisková organizace vyvíjející standardy pro e-business.

**PDA** (Personal Digital Assistant) – Mobilní zařízení nahrazující osobní počítač.

**PHP** (Personal Home Page) – Osobní domácí stránky. Hypertextový preprocesor, nástroj pro generování dynamických WWW stránek na principu scriptingu (obdoba ASP).

**PNG** (Portable Network Graphics) – Přenosná síťová grafika. Grafický formát, který by měl nahradit GIF.

**RosettaNet** – nezisková organizace vyvíjející otevřené e-business standardy.

**SAX** (Simple API for XML) – Jednoduché aplikační rozhraní pro XML založené na generování událostí.

**SOAP** (Simple Object Access Protocol) – Jednoduchý protokol pro přístup k objektům. Otevřený, rozšiřitelný způsob komunikace mezi aplikacemi pomocí XML zpráv přes HTTP, nezávisle na OS, objektovém modelu nebo jazyku.

**SGML** (Standard Generalized Markup Language) – Standardní obecný označovací jazyk. Systém označování částí dokumentů vsunutými údaji, standard dokumentů (ISO 8879 z r. 1986). Velmi obecný a komplexní jazyk.

**SMTP** (Simple Mail Transfer Protocol) – Jednoduchý protokol elektronické pošty. Standardní protokol na síti TCP/IP.

**UBL** (Universal Business Language) – průmyslový standard vyvíjený organizací OASIS, který definuje standardizované obchodní XML dokumenty (objednávky, faktury) příslušnými XML schémata.

**UN/EDIFACT** (United Nations/Electronic Data Interchange for Administration, Commerce, and Transport) – Elektronická výměna dat pro státní správu, obchod a dopravu. Doporučení v rámci Evropské komise.

**URI** (Uniform Resource Identifier) – Nejobecnější tvar identifikátoru nějakého datového zdroje. Existují i konkrétnější identifikátory a to URN (Uniform Resource Name) a URL (Uniform Resource Locator). Zatímco u URN nezáleží na fyzickém umístění informací, URL je úzce spjata s konkrétním uložením zdroje. Příklad URN: urn:ISBN:0-395-36341-1. Příklad URL: <http://www.w3c.org>. Všechny tyto identifikátory musí být celosvětově jednoznačné.

**WAV** (WAVE) – Vlna. Souborový typ a přípona souborů obsahujících zvukový záznam.

**W3C** (World Wide Web Consortium) – Konsorcium pro WWW. Organizace zabývající se standardy pro WWW.

**WM** (Workflow Management) – Management plánování toku práce. Zajišťuje automatizaci a správu toku práce.

**WML** (Wireless Markup Language) – Bezdrátový vyznačovací jazyk. Jazyk používající vyznačovací značky (tags) podobně jako HTML.

**xCBL** (XML Common Business Library) – Standard založený na XML, který se snaží o vývoj robustních, znovupoužitelných XML dokumentů pro elektronický obchod.

**XML** (eXtensible Markup Language) – Rozšiřitelný značovací jazyk. Platformově nezávislý formát pro publikování a výměnu dat a dokumentů koncepčně podobný SGML.

**XML-RPC** (XML – Remote Procedure Call) – sada implementací, která umožňuje aplikacím běžícím na různých operačních systémech, volat procedury prostřednictvím Internetu.

**XSL** (eXtensible Stylesheet Language) – Jazyk XSL lze použít pro tvorbu tzv. Stylesheetů nebo-li transformačních stylů. Jazyk XSL, oproti CSS, nabízí mnohem širší možnosti použití.

# Použité zdroje

## Literatura

- [1] Machková, Hana: Mezinárodní obchodní operace, Grada Publishing, Praha 2003, ISBN: 8024706865
- [2] Kosek, Jiří: XML pro každého podrobný průvodce, Grada Publishing, Praha 2000, ISBN: 8071698601
- [3] Kotler, Philip: Marketing od A do Z, Management Press, Praha 2003, ISBN: 8072610821
- [4] Bredly, Neil: XML, kompletní průvodce, Grada 2000, ISBN: 8071699497
- [5] Laurent St. Simon: Tvorba internetových aplikací v XML, Computer Press 1999, ISBN: 8072261703
- [6] J. Young, Michael: XML krok za krokem, IDnes 2002, ISBN: 8086593282
- [7] Elliotte Rusty Harold, W. Scott Means: XML in the nutshell second edition, O'Reilly 2002, ISBN 0596002920
- [8] McLaughlin, Brett: Java & XML, second edition, O'Reilly 2001, ISBN 0-596-00197-5
- [9] Erik T. Ray: Learning XML, second edition, O'Reilly 2001, ISBN: 0596004206
- [10] Holzner, Steven: Teach Yourself XML in 21 Days, third edition, Sams Publishing 2003, ISBN: 0672325764
- [11] Tidwell, Doug: XML programming in Java technology tutorial (1, 2, 3), [ibm.com/developerWorks](http://ibm.com/developerWorks)
- [12] Ennsner, Luis a kol.: Using XML for B2B and B2C Applications, IBM Redbooks tutorials, 2000

## Internet

- [13] <http://www.w3c.org>
- [14] <http://www.xml.com>
- [15] <http://www.xml.cz>

- [16] <http://www.interval.cz>
- [17] <http://wikipedia.org>
- [18] <http://ebxml.org>
- [19] <http://www.ebxml.eu.org/>
- [20] <http://www.webservices.org>
- [21] <http://www.w3schools.com>
- [22] <http://www.systemonline.cz/site/e-business/9horovcak.htm>
- [23] <http://www.ics.muni.cz/cgi-bin/toISO88592.eng/bulletin/issues/vol13num03/kuba2/kuba2.html>
- [24] <http://www.linuxzone.cz/index.phtml?ids=2&idc=343>
- [25] <http://www.softwareag.com/xml/>
- [26] <http://interval.cz/clanek.asp?article=3214>
- [27] <http://badame.vse.cz/izi238/prednasky.html>
- [28] <http://interval.cz/clanek.asp?article=1156>
- [29] <http://www.javaworld.com/javaworld/javaone00/j1-00-ebxml.html>
- [30] <http://www.idealliance.org/papers/xml2001papers/tm/web/05-02-03/05-02-03.htm>
- [31] <http://tech.irt.org/articles/js215/>
- [32] <http://interval.cz/clanek.asp?article=3214>
- [33] <http://xml.coverpages.org/cxml.html>
- [34] <http://webswell.com/products/faq/>
- [35] <http://www.datis.cd rail.cz/Black/Scripts/ZKRATKY/slovník.asp>
- [36] <http://www.ebay.com>
- [37] <http://www.jdom.org/>
- [38] <http://www.opensource.org/>
- [39] <http://java.sun.com/xml/jaxp/index.jsp>

[40] <http://www.saxproject.org>

[41] <http://cocoon.apache.org>

[42] <http://www.xcbl.org>

[43] <http://www.cxml.org>

[44] <http://www.oasis-open.org/committees/ubl>