# Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury pod vedením vedoucího a konzultanta. Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména §60 (školní dílo) a §35 (o nevýdělečném užití díla k vnitřní potřebě školy).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé práce a prohlašuji, že souhlasím s případným užitím mé práce (prodej, zapůjčení apod.)

Jsem si vědom toho, že užití své diplomové práce či poskytnutí licence k jejímu užití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do její skutečné výše).

Po pěti letech si mohu tuto práci vyžádat v Univerzitní knihovně TU v Liberci, kde je uložena, a tím výše uvedená omezení vůči mé osobě končí.

V Liberci dne 5. ledna 2004 Podpis. . . . . .

# Resumé

Cílem této práce je upozornit na potenciál této poměrně nové a rychle se rozvíjející technologie. Tato práce se zaměřuje zejména na praktické postupy a vytváření distribuovaných aplikací. Součástí práce je implementace řešení. V první části je vysvětlen účel webových služeb a popsány základy a význam všech technologií, které vytváří architekturu webových služeb. Druhá část této práce je zaměřena na návrh a implementaci webových služeb v dodavatelsko-odběratelském řetězci. Popisované zásady a postupy při vytváření webových služeb jsou provázeny ukázkami z XML souborů, které tvoří podstatu implementovaného řešení. Přestože je řešení vytvořeno pro studijní účely a dodavatelsko-odběratelských řetězec tvoří fiktivní subjekty, je aplikační model reálný a přístupný na internetu.

# Abstract

The thesis's goal is to draw attention to a potential of this quite new and quickly developing technology. Primarily this work focuses practical processes and development of distributed applications. The first part of the work explains the purpose of Web services and describes essentials and sense of all technologies creating Web services architecture. The second part focuses on design and Web services implementation in Supply chain. Principles and practices that are described are accompanied by samples from XML files, that formed the keystone of implemented solution. Although this solution is created for educational purposes and Supply chain forms fictitious subjects, application model is real and available on the internet.

# Obsah

# Seznam obrázků



# Seznam tabulek

# Seznam výpisů ze zdrojových textů

# Seznam zkratek

| | |
|---|---|
| API | Application Programming Interface |
| apod. | a podobně |
| atd. | a tak dále |
| COM | Component Object Model |
| CORBA | Common Object RequestBroker Architecture |
| CRM | Customer Relationship Management |
| DCE | Distributed Computing Environment |
| DCOM | Distributed Component Object Model |
| Disco | SOAP Discovery |
| EDI | Electronic Data Interchange |
| FTP | File Transfer Protocol |
| GXA | Global XML Web Services Architecture |
| HTML | Hypertext Markup Language |
| HTTP | HyperText Transfer Protocol |
| IDL | Interface Definition Language |
| IIOP | Internet Inter-Orb Protocol |
| J2EE | Java 2 Platform Enterprise Edition |
| MQ | Message Queue |
| OASIS | The Organization for the Advancement of Structured Information Systems |
| PDA | Personal Digital Assistant |
| RMI | Remote Method Invocation |
| RPC | Remote Procedure Call |
| SMTP | Simple Mail Transfer Protocol |
| SOAP | Simple Object Access Protocol |
| SSL | Secure Socket Layer |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| TLS | Transport Layer Security |
| tModel | Technolology Model |
| tj. | to je |
| UDDI | Universal Discovery Description and Integration |
| W3C | Worldwide Web Consortium |
| WSDL | Web Service Description Language |

| | |
|---|---|
| XML | Extended Markup Language |
| XSLT | Extensible Stylesheet Language Transformations |

# Značení

Pro lepší čitelnost budou v textu použity různé druhy písma.

- Zdrojové texty v XML
  - názvy elementů
  - názvy atributů
  - hodnoty atributů
  - hodnoty elementů
- *Názvy tříd a metod*
- `Názvy elementů v textu`

# 1 Úvod

V současné době se stáváme svědky rozvoje elektronického obchodování. Obchodování na internetu, tak jak ho známe dnes, se točí kolem webových stránek. Webové stránky prezentují statické informace uživateli. Pokud jsou propojeny s existujícími systémy mohou uskutečňovat jednoduché transakce a zajišťovat v omezené míře obchodování. Aby podniky mohly využít možnosti, které s sebou internet přináší, webové stránky se musí změnit. Musí se naučit komunikovat mezi sebou samými, spolu s existujícími systémy a aplikacemi.

Podnikové aplikace pro e-business využívají v současné době jen malou část služeb, které poskytuje internet. Různé podnikové informační systémy a mainframové aplikace mají často problémy se vzájemnou interakcí, takže jsou pro e-business téměř nepoužitelné. Integrace podnikových systémů je náročná a často se stává, že náklady na integraci přesahují výnosy.

Výše uvedené problémy řeší nová a rychle se rozvíjející technologie webových služeb. Cílem této práce je popsat její možnosti a využití, které demonstruji na konkrétním řešení.

Práce je členěná do kapitol, které se uvedené problematice věnují od teoretických základů až po praktický návrh a jednoduché řešení.

V úvodu této práce rozebírám mezery současného internetu, především v přístupu k informacím. Na jednoduchém příkladě vysvětluji myšlenku a princip fungování webových služeb, které nedostatky dnešního internetu v přístupu k informacím řeší.

Druhá kapitola je zaměřena na základy technologie, kde jsou podrobně popsány jednotlivé standardy a specifikace, které vytváří celou architekturu webových služeb.

Ve třetí a čtvrté kapitole se zaměřuji na praktické nasazení webových služeb. Hlavní pozornost věnuji teoretickým základům, analýze, návrhu řešení až po implementaci. Jednotlivé části jsou doplněny praktickými ukázkami z XML souborů, které tvoří základní rámec ukázkového řešení.

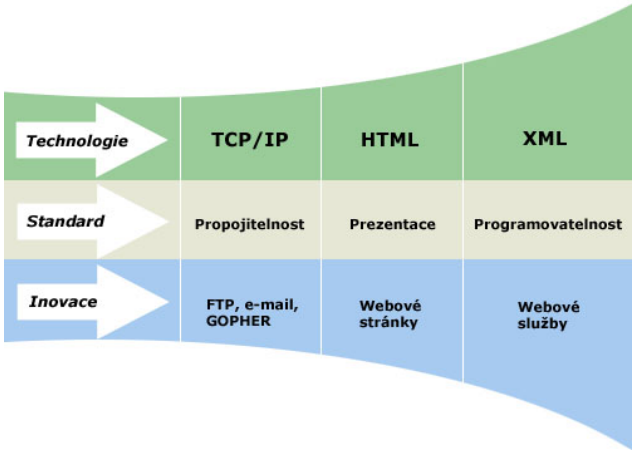
V závěrečné části této práce se zabývám významem webových služeb pro e-business a věnuji se předpokládanému vývoji a budoucnosti této rychle se rozvíjející technologie.

Literatura a informační zdroje použité v této práci jsou uvedeny v příloze. Většina informací byla získána z internetu, odborných článků a stránek různých organizací, které se používanými technologiemi zabývají.

## 1.1 Vznik webových služeb

Internet v té podobě, jak ho známe dnes, je založen na sérii postupných inovací, řízených technologickými standardy, vytvořenými většinou v průběhu posledních deseti let. Jak vyplývá z obrázku (viz. Obr. 1-1), je každá inovace založena na úspěchu předchozího zlepšení, což nás nakonec přivádí až k možnostem vytvářet síťové aplikace pro zpracování dat přímo na síti.

**Obr. 1-1:** Vznik současného internetu jako série různých technologických zlepšení




***Zdroj**: Brian E. Travis, XML a SOAP, str. 133*

V roce 1980 byl vyvinut protokol TCP/IP, který umožňuje propojovat různé soukromé sítě a zasílat mezi nimi pakety po veřejných sítích. Jakmile bylo možné propojovat jednotlivé sítě mezi sebou, aplikace začaly využívat výhody z tohoto faktu plynoucí. Již první technologie využívající protokoly TCP/IP (protokol FTP a e-mail) umožňovaly vzájemnou komunikaci mezi lidmi. Tím začala revoluce i ve způsobu obchodování [1].

I přes všechny své přednosti však FTP ani e-mail nenabízely způsob, jakým bychom mohli na informace nahlížet konzistentním způsobem přes důsledně jednotné rozhraní. Jazyk HTML byl vytvořen jako jednoduchý nástroj pro značkování dokumentů, které jsou následně zveřejňovány na webu a prohlíženy ve speciálních programech. Jeden z tagů tohoto jazyka (tag A) umožnil plně využít výhody veřejně propojených sítí a nabídl jednoduchý a účinný prostředek pro přesun z jedné stránky do jiné [1].

Sdí1ení navzájem propojených dokumentů po síti znamenalo převrat ve způsobu nahlížení na informace, protože uživatelé nyní mohli společně využívat údaje nezávislé na použitém operačním systému nebo konkrétní aplikaci. Každý si nyní mohl prohlížet celou síť z jedné jediné aplikace [1].

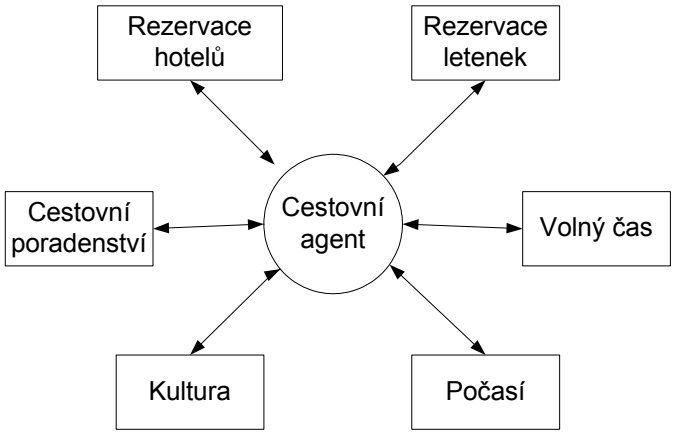
HTML je integrální technologií pro výměnu informací mezi lidmi, ale ztrácí své přednosti, jakmile začneme uvažovat i o výměně informací mezi počítači. HTML je především navržen pro plošnou prezentaci informací, která je určena k prohlížení. Počítače prozatím oči nemají, takže jim HTML moc nevyhovuje. HTML nás dále nutí rozprostřít informace tak, aby vyplnily plochu obrazovky (ale ne víc), a ponechává příliš mnoho prostoru pro naši schopnost chápat nejednoznačné významy [1].

## 1.2 Základní myšlenka webových služeb

Podívejme se například na pracovní zvyklosti agenta cestovní kanceláře. Jeho úkolem je rozeznat, co jeho klienti chtějí v rámci dovolené dělat a nabídnout jim takový balíček služeb, který si od něj koupí. Pokud požádám svou agentku, aby mi našla místo pro dovolenou kde "bude levně, horko a bude to i pro rodiny s dětmi", dostala v této větě množství údajů, aby mohla začít s hledáním. Pravděpodobně budu muset upřesnit, co já chápu jako "levnou dovolenou“, co pro mě znamená „horko“ atd. Dále se mě nejspíše zeptá na moji rodinu - například jestli už děti odrostly nebo zda to nejsou náhodou batolata a poté se pustí do práce. Nejdříve přezkoumá ta místa ve světě/kde bude horko právě v době, kdy se já chystám na dovolenou. Pak si porovná ceny letenek, ceny v hotelech, poplatky za pronájem auta a spoustu dalších faktorů, které nějak ovlivňují výslednou cenu. K porovnání může využít třeba tištěné brožurky, které hotely a cestovní kanceláře nabízejí zdarma, nebo může vznášet dotazy na nejrůznějších webových sídlech, která se specializují na záležitosti spojené s cestováním.

Z obrázku vyplývá (viz. Obr. 1-2), že moje agentka shromáždí spoustu informací a v nich takové kombinace služeb, které odpovídají mým potřebám. Nachází se vlastně přímo ve středu informačního vesmíru.

**Obr. 1-2:** Cestovní agent shromažďuje informace z mnoha zdrojů.




***Zdroj**: Brian E. Travis, XML a SOAP, str. 135.*

Po přezkoumání různých zdrojů informací začne fáze rozhodování. Například na Kajmanských ostrovech může teplota dosáhnout až 30 stupňů ve stínu, ale hotely jsou velmi drahé a z Denveru je jen omezené letecké spojení. Na Svatém Tomáši bude sice jen 28 stupňů, ale celkově je tam mnohem levněji a lepší je i spojení. Jenže zase děti by tam neměly co dělat. Bahamy jsou jen o něco málo dražší a mohlo by tam být tepleji, ale děti si tu užijí do sytosti. Je tedy jasné, že bude muset udělat pár kompromisů a svoje požadavky trochu uzpůsobit. Nakonec z té sumy informací agentka vypracuje několik návrhů a ty mi předloží.

Moje agentka je "jen" člověk, a tak může najednou dělat jen pár věcí. Samozřejmě má také jiné zákazníky, nemůže se zrovna mně věnovat celý rok. Díky těmto omezením nemůže prověřit všechny možné příležitosti pro mou dovolenou, ale jen jejich menší část. Dá se předpokládat, že existuje ideální dovolená, která splňuje všechny mé požadavky. Jenže pokud tato informace nedorazí na patřičné místo (do kanceláře mé agentky) nebude brána do úvahy a já přijdu o pěknou dovolenou.

Tady je právě prostor pro počítače, které by mohly během relativně krátké chvíle vyhledat a zhodnotit takové množství informací, které by člověk mohl zpracovávat třeba léta. Bylo by to moc pěkné, ale asi by to příliš nefungovalo. Problém je totiž, ve formátu dostupných dat.

Jestliže moje agentka získává informace na webových stránkách, je tato informace stahována ve formátu HTML a ona si je čte v prohlížeči. Většina webových stránek tohoto typu (obchodní nabídky) je vytvářena až na požádání nějakou aplikací umístěnou ve střední vrstvě daného

webového sídla. V případě, že se dotáže na stránce nějakého hotelu, zda v něm budou v požadovaném termínu volná místa, vznesou příslušné programy dotaz do databáze obsahující tabulky s daty o obsazení pokojů a ze získaných údajů pro ni vytvoří stránku HTML.

Pokud bychom měli vytvořit software, který by k těmto informacím přistupoval stejným způsobem, jako to dělá moje agentka, měl by počítač velké problémy. Musel by totiž prozkoumat staženou stránku HTML jako jednorozměrný řetězec znaků a ztratil by tak naši možnost dívat sena tyto informace plošně a vidět v nich tak třeba fotografii dané lokality nebo text rozvádějící určitá základní fakta. Jestliže se vy podíváte na stránku HTML a uvidíte v ní například „33°C", tak víte, že to je údaj o teplotě a že taková teplota v létě spoustě lidí vyhovuje. Počítač ale vidí čtyři znaky a o jejich lidské interpretaci mu není známo vůbec nic.

Kromě stránek HTML to můžou pochopitelně být dokumenty nejrůznějších formátů, více či méně podporované, svázané s určitými platformami či technologiemi, takže jen jejich samotné zobrazení může být problém.

Vraťme se k hypotetické stránce určitého hotelu. Víme, že se někde v ní nacházejí velmi hodnotné údaje. Aplikace na straně serveru tato data "zlidšťují“, aby je čtenáři mohli snadno pochopit. Určitě by bylo lepší, kdyby si počítače mohly podobná data mezi sebou vyměňovat a pracovat s nimi, namísto jejich převodu do dvou rozměrů. Kdybychom mohli počítači v hotelu sdělit, že získaná data bude zpracovávat počítač nikoli člověk, mohl by tento počítač zaslat zpět do mé aplikace data, která by se dala mnohem snadněji zpracovat.

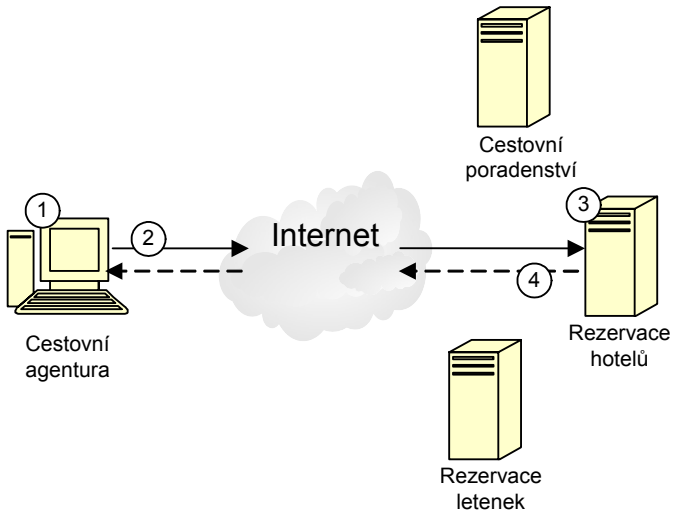
Například na požadavek klientské aplikace na průměrnou teplotu v určité lokalitě může webová služba hotelu vrátit tuto odpověď:

**<prumer-teplota stupnice="celsius">28</prumer-teplota>**

Vrácené informace jsou v textové podobě, navíc snadno čitelné. Každý jistě z ukázky rozpozná, že vrácená hodnota je 28, která je ve stupních Celsia. Tento formát se nazývá XML (kapitola 2.1) a v současnosti je to standardní formát pro výměnu dat.

Program zajišťující tento typ komunikace mezi počítači se jmenuje webová služba a jde o další významnou inovaci v rámci infrastruktury webu. K výměně a sdílení informací webové služby používají výhradně jazyk XML.

**Obr. 1-3**: Základní myšlenka webových služeb.




***Zdroj:*** *vlastní*

**Tab. 1-1:** Popis jednotlivých fází z obrázku 1-3.

| Fáze v obrázku | Popis |
|---|---|
| **1** | Program na straně klienta vytvoří zprávu ve formátu XML obsahující údaje potřebné pro vzdálené volání metody webové služby. |
| **2** | Zpráva je odeslána protokolem HTTP. |
| **3** | Webová služba obdrží zprávu. Ze zprávy získá potřebné parametry a na jejich základě provede požadovanou operaci. Výsledek operace je převeden zpět do zprávy ve formátu XML. |
| **4** | Zpráva je odeslána protokolem HTTP zpět klientovi. |

***Zdroj:*** *vlastní*

V předchozí tabulce (viz. Tab. 1-1) je jednoduchým způsobem vysvětlen princip výměny informací mezi klientem a webovou službou. V následujících kapitolách bude tento postup upřesněn a detailně rozebrán.

Na webové služby se můžeme dívat jako na aplikace, které vašim programům zpřístupňují data na webu dostupná, takže k nim vaše aplikace může přistupovat, jako by byly uloženy někde v lokálním souboru. Tato schopnost programovat web bude základem našeho používání internetu v budoucích letech.

V předchozím příkladu je velmi jednoduše a srozumitelně vysvětlena základní myšlenka webových služeb. S určitým zjednodušením lze říci, že webové služby zajišťují vlastně komunikaci mezi jednotlivými počítači.

# 2 Základy technologie a standardy

Vzájemná komunikace webových služeb je založená na dodržení tří základních standardů. Jsou jimi normy pro popis webových služeb WSDL, protokol vzájemné komunikace objektů SOAP založený na textovém formátu XML a seznam webových služeb UDDI. Tyto standardy jsou vysvětleny v následujících kapitolách.

## 2.1 XML – podstata webových služeb

Dnešní doba přeje komunikaci. Komunikace není nic jiného, než výměna informací. V dnešním globálním světě není možné pro výměnu dat používat nějaké proprietární formáty, které jsou svázány s konkrétním softwarem nebo hardwarem. Nesluší se posílat informace ve wordovém formátu DOC, protože někdo s unixovým počítačem si je těžko přečte. Centrále nadnárodní společnosti asi nebudeme výroční zprávu české pobočky posílat ve formátu T602, protože ve své americké verzi kancelářského balíku si ji nikdo nepřečte. Je potřeba používat nějaký jednoduchý otevřený formát, který není úzce svázán s nějakou platformou nebo proprietární technologií.

Takovým formátem je XML. Otevřený formát je to proto, že jeho specifikace je každému zdarma k dispozici na serveru konsorcia W3C, které se stará i o mnoho dalších technologií souvisejících s webem. Každý tak může bez problémů do svých aplikací implementovat podporu XML. To představuje velký rozdíl oproti firemním formátům, k nimž není k dispozici žádná dokumentace a navíc se jedná v porovnání s XML o velice složité formáty, často binární [2].

Práci s XML usnadňuje i to, že celý formát je založen na obyčejném textu. I když pro většinu lidí zůstane kód XML skryt a budou ho používat pouze aplikace pro vzájemnou komunikaci, není problém kdykoliv otevřít XML dokument v libovolném textovém editoru a pár potřebných úprav provést ručně. Použití textového formátu může někomu připadat jako zbytečné plýtvání místem. Dnes se však mnohem větší důraz klade na srozumitelnost a snadnou práci s daty. Jestli ušetříme pár kilobajtů paměti, již nikoho příliš netrápí [2].

Patronát nad pravidly definujícími XML (a jiné jazyky používané v prostředí Internetu) má organizace W3C [14], což je sdružení firem a uživatelů, které se snaží o normalizaci na Internetu.

Brzy se přišlo na to, že XML není jen vynikajícím prostředkem pro popis dat, ale má i širší použití. Nezávislost na platformě dělá z XML vynikající prostředek ke spolupráci mezi aplikacemi nacházející se na různých programovacích platformách a operačních systémech. XML se stal univerzálním jazykem pro výměnu informací a v poslední době pak k integraci aplikací, ke kterému slouží právě webové služby.

## 2.2 Webové služby

### 2.2.1 Definice

Webové služby jsou menší znovupoužitelné aplikace používající XML. Umožňují přenos dat prostřednictvím internetu (intranetu) mezi jinak nepropojenými zdroji, které jsou je schopny poskytovat nebo využívat, bez ohledu na to, kde jsou umístěny nebo jak byly implementovány, například [25]:

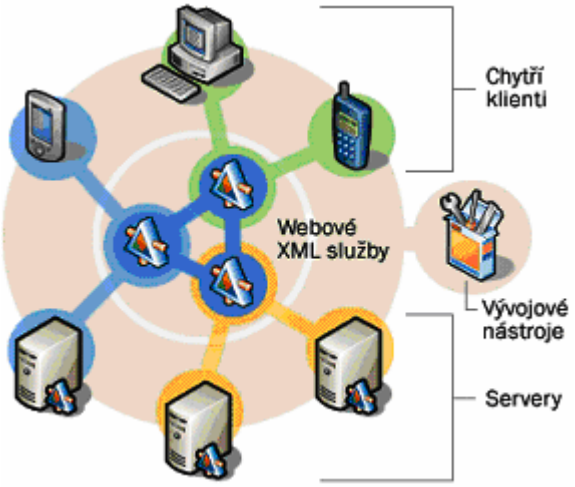
**Klient-klient:** "chytří" klienti mohou poskytovat nebo využívat webové XML služby, což umožňuje sdílet data kdekoli a kdykoli.

**Klient-server:** webové XML služby umožňují sdílet data ze serverové aplikace na počítači nebo mobilním zařízení přes internet.

**Server-server:** webové XML služby poskytují společné rozhraní mezi již existujícími aplikacemi běžícími na nezávislých serverech.

**Služba-služba:** webové XML služby mohou na sebe navazovat a tak vytvořit komplexnější datovou operaci.

**Obr. 2-1:** Webové služby - propojení zdrojů




***Zdroj:*** *Microsoft ČR [25]*

Existuje pravděpodobně tolik definicí webových služeb XML, kolik je podniků, které je staví. Většina těchto definicí však má společné následující:

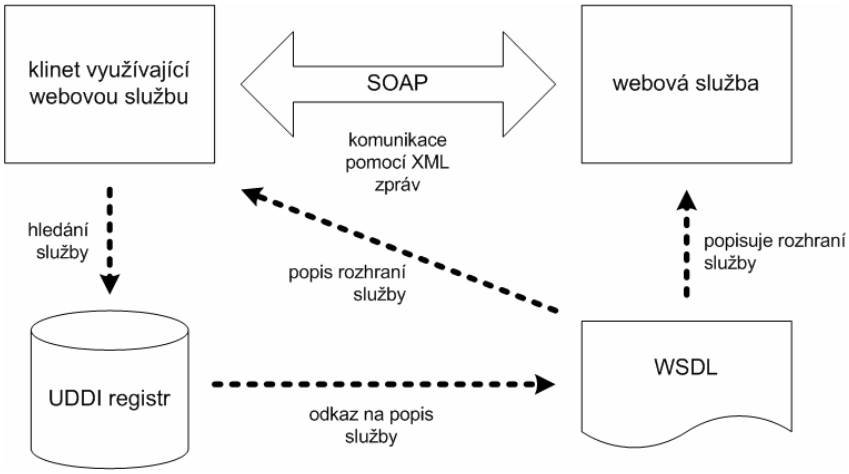
- Webové služby XML poskytují webovým uživatelům užitečné funkce prostřednictvím standardního webového protokolu. Ve většině případů se používá protokol SOAP (kapitola 2.3).
- Webové služby XML poskytují způsob, jak dostatečně podrobně popsat svoje rozhraní, aby uživatelé mohli stavět klientské aplikace, které s nimi mohou komunikovat. Tento popis je obvykle poskytován ve formě dokumentu XML nazvaného Web Services Description Language (WSDL) (kapitola 2.4).
- Webové služby XML jsou registrované, takže potenciální uživatelé je mohou snadno nalézt. To zajišťuje specifikace Universal Discovery Description and Integration (UDDI) (kapitola 2.5).

### 2.2.2 Vztahy mezi technologiemi

Vzájemné vztahy mezi těmito třemi technologiemi jsou zachycené na následujícím obrázku (Obr. 1-1). Ke každé webové službě by měl být k dispozici její formální popis v jazyce WSDL. Z tohoto popisu již jde automaticky vygenerovat požadavek SOAP. Ve větších systémech nebo přímo v otevřeném prostředí Internetu se popis služby může zaregistrovat do UDDI registru. Ten slouží jako jakýsi telefonní seznam („zlaté stránky“), který umožňuje vyhledávání služeb s určitými parametry.

Klient, který chce využít webovou službu, získá buď přes UDDI, nebo přímo její popis. Z něj je jasné, jakou strukturu má mít zpráva SOAP a kam se má webové službě poslat, aby ji rozpoznala.

**Obr. 2-2:** Vztah tří základních technologií (SOAP, WSDL a UDDI) webových služeb




***Zdroj:*** *Jiří Košek [26]*

### 2.2.3 Komunikace mezi klientem a webovou službou

V následujících bodech je shrnut postup komunikace mezi klientem a webovou službou [3].

1. Na UDDI webové službě zjistíme, zda vůbec někde existuje služba, o kterou máme zájem. Například může jít o informace týkající se burzovních indexů. Výsledkem je seznam serverů nebo přímo služeb na konkrétních serverech ve formátu URL odkazu.

2. V případě, že odpověď z UDDI obsahuje odkazy na webové servery, musíme zaslat nový dotaz na Disco dokument na konkrétním serveru. Výsledkem je odpověď ve formátu XML s odkazem na jednotlivé služby implementované serverem. Nyní již budeme mít přesně jasno, jak se služba jmenuje a kde je umístěna.

3. Zbývá požádat službu samotnou, aby vrátila popis svých metod ve formátu dokumentu WSDL. Po jeho přijetí je nejčastěji vygenerován proxy objekt, sloužící k jednoduché komunikaci se službou. V každém případě po dokončení třetího kroku jsou k dispozici naprosto všechny informace potřebné pro vytvoření dotazu a přijetí odpovědi.

4. Jednotlivé dotazy jsou formátovány jako SOAP dokumenty nebo HTTP-GET nebo HTTP-POST operace odesílané webové službě. Ta po jejich zpracování vrací XML nebo SOAP dokument s výsledky. Díky proxy objektům, vygenerovaným v kroku tři, není nutné podrobně znát specifikaci protokolu XML nebo SOAP. O konkrétní formulaci dotazu a interpretaci odpovědí se v postarají právě objekty proxy.

### 2.2.4 Architektura webových služeb

Potenciál webových služeb je samozřejmě obrovský. Zejména v rámci firem může jejich využití znamenat velký přínos. Dosud totiž bylo velmi obtížné integrovat jednotlivé aplikace, neboť byly často naprogramovány v různých jazycích a nebo běžely na různých platformách, a jejich vzájemné propojení nefungovalo vždy zcela spolehlivě.

Jednou z hlavních výhod architektury webových služeb je to, že programy mohou být napsány v různých jazycích na různých platformách a že spolu navzájem komunikují standardním způsobem (XML).

Ti z vás, kteří pracujete v oboru delší dobu si nyní jistě říkáte, že jste podobné prohlášení slyšeli o komponentových technologiích jako CORBA, DCOM, RMI, nebo IIOP. Webové služby jsou však již ve své podstatě naprosto rozdílné.

#### *2.2.4.1 Jednoduchost a použití standardů*

Prvním rozdílem je skutečnost, že SOAP je podstatně jednodušší než dřívější přístupy, takže překážky zavádění standardům vyhovující implementaci SOAP jsou podstatně menší.

Výše zmíněné komponentové technologie mají jednu velkou nevýhodu, pokud jsou provozovány v otevřeném heterogenním prostředí Internetu. Všechny jsou založeny na proprietárních protokolech a platformách, na kterých jsou provozovány. Jejich nasazení významně limituje oblast klientů, kteří jejich služby na vzdálených počítačích mohou využívat.

Další podstatnou výhodou webových služeb proti předcházejícím pokusům je to, že pracují se standardními webovými protokoly - XML, HTTP, TCP/IP nebo SMTP. Velké množství firem již má vybudovanou webovou infrastrukturu a má lidi se znalostmi a zkušenostmi s její údržbou. Náklady na zavedení webových služeb jsou tak podstatně nižší než u předcházejících technologií.

#### 2.2.4.2 Popis funkcionality

Druhá zásadní změna nastala v oblasti popisu funkcionality služby a předávání dat mezi klientem a serverem. Nemalým problémem v otevřeném prostředí je potřebná znalost komunikačních rozhraní a datových typů. Tvůrce klientské aplikace musí být dopředu obeznámen se všemi detaily. Bez nich se jednoduše "nedomluví". Za přispění standardních protokolů webové služby samy poskytují popis své funkcionality. Součástí je i charakteristika datových typů pro předávání parametrů a návratových hodnot jednotlivých funkcí. Klient nyní nemusí dopředu znát detailní popis komunikačního rozhraní, ale může si ho kdykoliv zjistit.

#### 2.2.4.3 Formát přenášených dat

Klíčovou otázkou je také způsob předávání samotných dat. Binární forma nemusí být, na všech různých operačních a výpočetních systémech identická. Webové služby opět využívají jazyk XML, umožňující předat data zcela unifikovaně, platformově nezávislým způsobem.

Protože data a dotazy (volání) jsou přenášeny protokolem HTTP ve formátu XML (protokol SOAP), který je textové podoby a protokol HTTP je běžně na portu 80 povolený, je tím navíc zajištěn bezproblémový průchod přes všechny aktivní prvky, jako například firewally, které se vyskytují v kterékoliv počítačové síti. SOAP je standardizovaným protokolem, který se stává dostupným na všech výpočetních systémech.

První webové služby měly tendenci být informačními zdroji, které mohou být snadno začleněny do aplikací - ceny akcií, předpovědi počasí, sportovní výsledky atd. Je snadné si představit celou třídu aplikací, které mohou být postaveny k analýze a shromažďování žádaných informací a jejich prezentaci různými způsoby; můžete mít například tabulku aplikace Microsoft Excel, která shrnuje celou vaši finanční situaci - akcie, spoření, bankovní účty, půjčky atd. Pokud jsou tyto informace dostupné prostřednictvím webových služeb, může je Excel neustále aktualizovat. Některé informace mohou být poskytovány zdarma, jiné můžete získat pouze, když si předplatíte určitou službu. Většina těchto informací je již na webu dostupná, ale webové služby k nim zajistí snadnější a spolehlivější programovaný přístup.

Nabídka existujících aplikací ve formě webových služeb umožní uživatelům stavět nové, výkonnější aplikace využívající webové služby jako stavební bloky. Uživatel může například vyvinout nákupní aplikaci k automatickému získání cenových informací od různých prodejců, která uživateli umožňuje vybrat prodejce, předat objednávku a pak sledovat dodávku až do jejího přijetí.

Prodejní aplikace může zase navíc k nabídce služeb na webu využít webové služby ke kontrole úvěru zákazníka, provádět příjem peněz z účtu zákazníka a zajistit dodávku zboží přes dodávkovou službu.

## 2.3 SOAP

### 2.3.1 Specifikace

SOAP je protokol pro posílání zpráv XML a je základem webových služeb. Ostatní standardy jako WSDL a UDDI vznikly až později po uvedení SOAP a jen dále rozšiřují jeho možnosti a snadnost použití. SOAP umožňuje zaslání XML zprávy mezi dvěma aplikacemi a pracuje tedy na principu peer-to-peer. Zpráva je jednosměrný přenos informace od odesílatele k příjemci, ale díky kombinování několika zpráv můžeme pomocí SOAP snadno implementovat běžné komunikační scénáře.

Nejčastěji se SOAP používá jako náhrada vzdáleného volání procedur (RPC), tedy v modelu požadavek/odpověď. Jedna aplikace pošle v XML zprávě požadavek druhé aplikaci, ta požadavek obslouží a výsledek zašle jako druhou zprávu zpět původnímu iniciátorovi komunikace. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy přes HTTP přijde zpráva SOAP, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď.

SOAP rovněž podporuje aplikace ve stylu dokumentu, kde zpráva SOAP je pouze obalem kolem dokumentu XML. Aplikace SOAP ve formě dokumentu jsou velmi pružné a mnoho nových webových služeb využívá tuto pružnost ke stavbě služeb, které by bylo obtížné implementovat pomocí RPC.

Poslední volitelná část specifikace SOAP definuje, jak má vypadat zpráva HTTP, která obsahuje zprávu SOAP. Tato vazba na HTTP je důležitá, neboť protokol HTTP má širokou podporu v různých aplikacích. Vazba HTTP je volitelná, ale téměř všechny implementace SOAP ji podporují, neboť to je jediný standardizovaný protokol pro SOAP. Z tohoto důvodu vznikla obecně mylná představa, že SOAP vyžaduje HTTP. Některé implementace podporují přenosy MQ Series, SMTP nebo TCP/IP, ale většina současných webových služeb používá HTTP, neboť je všudypřítomný. HTTP je základní protokol webu a většina organizací má síťovou infrastrukturu, která podporuje HTTP a lidi, kteří s tímto protokolem umějí pracovat. V současné době je také pro HTTP snadno dostupná infrastruktura zabezpečení, monitorování a vyrovnávání zátěže.

### 2.3.2 Implementace

Hlavním zdrojem nedorozumění při zahájení práce se SOAP je rozdíl mezi specifikací SOAP a mnoha implementacemi specifikace SOAP. Většina lidí používajících SOAP nepíše zprávy SOAP přímo, ale používá k tvorbě a rozboru zpráv SOAP nástrojový kit SOAP (toolkit). Tyto toolkity obecně převádějí volání funkce z určitého jazyka do zprávy SOAP. Microsoft SOAP Toolkit 2.0 například převádí funkční volání COM do SOAP a Apache Toolkit převádí funkční volání JAVA do SOAP. Typy funkčních volání a datové typy podporovaných parametrů jsou u každé implementace SOAP odlišné a funkce, která pracuje s jedním toolkitem nemusí pracovat s jiným. Není to však omezení protokolu SOAP, nýbrž určité používané implementace.

Daleko nejdůležitější funkcí SOAP je možnost implementace na mnoha různých hardwarových a softwarových platformách. Znamená to, že protokol SOAP může být využit ke spojení neslučitelných systémů uvnitř i mimo vaši organizaci. V minulosti bylo uskutečněno mnoho pokusů o vytvoření obecného komunikačního protokolu, který by mohl být využit k integraci systémů, ale žádný z nich nedosáhl tak širokého uplatnění jako SOAP. Je tomu tak proto, protože SOAP je mnohem menší a jednodušší k implementaci než mnoho předchozích protokolů. Například CORBA vyžaduje k implementaci několik let, takže dosud bylo uvedeno jen několik implementací. SOAP však může na většinu těžké práce používat existující XML Parsers a knihovny HTTP, takže implementace SOAP může být dokončena řádově za měsíce. To je také důvod, proč je již dostupných více než 70 implementací SOAP [12]. SOAP samozřejmě neumí všechno jako DCE nebo CORBA, ale jednoduchost výměny funkcí způsobuje, že je SOAP tak snadno dostupný.

### 2.3.3 Struktura zprávy SOAP

Zpráva v SOAP je jednoduchý XML dokument, který má kořenový element `Envelope`. V této obálce jsou pak uzavřeny dva elementy `Header` (hlavička) a `Body` (tělo). Hlavička je přitom nepovinná a používá se pro přenos pomocných informací pro zpracování zprávy – například identifikaci uživatele, autentizační informace (jméno, heslo) apod.

O to nejdůležitější se stará tělo zprávy, v němž se přenášejí informace identifikující volanou službu a předávané parametry, resp. návratové hodnoty služby. SOAP používá jmenné prostory pro identifikování jednotlivých částí XML zprávy. Obálka, hlavičky a tělo zprávy patří do jmenného prostoru http://schemas.xmlsoap.org/soap/envelope/ ze specifikace W3C [4] .

**Obr. 2-3:** Struktura SOAP dokumentu


***Zdroj:*** *Brian E.Travis, XML a SOAP, str. 147.*

**Výpis 2-1:** Ukázka zprávy SOAP

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=“http://schemas.xmlsoap.org/soap/envelope/“
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
      <m:VratPrumerTeplotu xmlns:m="urn:diplomka:priklad:Pocasi">
        <misto>CCD</misto>
      </m: VratPrumerTeplotu >
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

***Zdroj:*** *Vlastní*

Předchozí ukázka (

Výpis **2-1**) navazuje na příklad z kapitoly 1.2 a zobrazuje velmi jednoduchý SOAP požadavek na zjištění průměrné teploty v přímořském letovisku. Ukázková zpráva pro jednoduchost neobsahuje hlavičku, ale pouze tělo. V něm je požadavek na vyvolání vzdálené funkce *VratPrumerTeplotu* s parametrem pojmenovaným *misto* s hodnotou *CCD* (fiktivní kód pro letovisko Střední Dalmácie v Chorvatsku).

Jak by mohla vypadat XML zpráva s výsledkem přibližuje následující ukázka (viz. Výpis 2-2).

**Výpis 2-2:** Ukázka zprávy s odpovědí

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
     SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
     <m:VratPrumerTeplotuOdpoved  xmlns:m="urn:diplomka:priklad:Pocasi">
```

```
      <prumer-teplota stupnice="celsius">28</prumer-teplota>
    </m:VratPrumerTeplotuOdpoved>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

***Zdroj:*** *Vlastní*

### 2.3.4 Zasílání SOAP zpráv přes protokol HTTP

1. Klient SOAP (nemusí to být tradiční klient, může se jednat o webový server, webovou aplikaci, ale také součást desktopu) vytváří dokument XML s údaji pro vzdálené volání metody objektu na externím systému. Vytvoří požadavek na server SOAP, zabalí XML dokument do obálky SOAP a vysílá ho jako požadavek HTTP POST.
2. Celá obálka je odeslaná klasickým připojením protokolu HTTP.
3. Příjmová aplikace - server SOAP - dostane zprávu. Touto aplikací je obyčejně webový server, který analyzuje došlou obálku, zavolá příslušný objekt a odevzdá mu přitom potřebné parametry, které přišly v dokumentu SOAP.
4. Objekt vykoná požadovanou operaci a vrátí získanou informaci serveru SOAP. Server SOAP zabalí odpověď do obálky SOAP.
5. Obálka je odeslaná zpět do počítače, odkud přišel požadavek. SOAP dokument je uschovaný pod hlavičkou HTTP.
6. Klient SOAP čeká na odpověď objektu. Když přijde, klient odstraní obálku a odešle dokument té aplikaci, která ho potřebuje.

Všudypřítomnost HTTP a jednoduchost SOAP vytváří ideální základnu k implementaci Webových služeb XML, které mohou být volány z téměř jakéhokoli prostředí. Další informace o SOAP lze nalézt ve specifikaci SOAP [4] od W3C.

## 2.4 WSDL

### 2.4.1 Definice

WSDL je zkratka pro Web Services Description Language (jazyk popisu webových služeb). Pro naše účely postačí, když řekneme, že soubor WSDL je dokument XML popisující sadu zpráv SOAP a způsob, jakým se zprávy vyměňují. Jinými slovy, WSDL je pro SOAP totéž, čím je IDL

pro CORBA nebo COM. Jelikož WSDL má formát XML, je snadno čitelný a upravitelný, ale většinou je generován a využíván softwarem.

Abychom si uvědomili hodnotu WSDL, představme si, že chceme začít volání metody SOAP, poskytované jedním z našich obchodních partnerů. Můžeme jej požádat o nějaké ukázkové zprávy SOAP a napsat aplikaci tak, aby vytvářela a přijímala zprávy, které vypadají jako tyto ukázky. Tento způsob však může být náchylný ke vzniku chyb. Vidíte například ID zákazníka s hodnotou 2837 a předpokládáte, že to je celé číslo (integer), ale ve skutečnosti je to řetězec. WSDL specifikuje, co musí obsahovat zpráva s požadavkem a jak má vypadat zpráva s odpovědí v jednoznačném zápisu.

Zápis (notace), který soubor WSDL používá k popisu formátů zpráv je založen na standardu XML Schema [6]. Znamená to, že je neutrální k programovacímu jazyku a je založen na standardech, takže je vhodný k popisu rozhraní webových služeb přístupných z celé řady platforem a programovacích jazyků. Kromě popisu obsahu zpráv definuje WSDL, kde je služba dostupná a jaký komunikační protokol je použit ke spojení se službou. Soubor WSDL tedy definuje vše potřebné k tomu, aby mohl být napsán program k práci s webovou službou. K dispozici je několik nástrojů ke čtení souboru WSDL a ke generování kódu vyžadovaného ke komunikaci s webovou službou.

Mnoho současných nástrojových kitů SOAP obsahuje nástroje ke generování souborů WSDL z existujících programovacích rozhraní. Existuje i několik nástrojů k přímému psaní WSDL, přesto není nástrojová podpora WSDL tak kompletní, jak by mohla být. Nemělo by trvat dlouho, aby nástroje k tvorbě souborů WSDL a následnému generování proxy objektů - podobně jako u nástrojů CORBA nebo COM IDL - se staly součástí většiny implementací SOAP. WSDL se tak stane preferovaným způsobem tvorby rozhraní SOAP pro webové služby.

### 2.4.2 Struktura WSDL

WSDL dokument definuje služby jako kolekci koncových bodů v síti nebo protokolů zpracovávajících zprávy. Operace a zprávy jsou popisovány na abstraktní úrovni a teprve poté jsou svázány s konkrétním síťovým protokolem a datovým formátem. To umožňuje snadné vytvoření popisu rozhraní, které nabízí jednu službu několika způsoby. V praxi WSDL popisy nejčastěji popisují služby, které si posílají zprávy pomocí formátu SOAP a protokolu HTTP.

WSDL jazyk je velice obecný, aby zachoval platformovou nezávislost. Popišme si, jak takový popis webové služby ve WSDL vypadá. WSDL dokument obsahuje popis jedné služby (klíčové

slovo `service`), což je největší jednotka. Jedna služba má jednu nebo více bran (`port`). Každá brána má vazbu (`binding`), což je způsob jak se daná brána volá (například SOAP přes HTTP), a nějakou přístupovou adresu (URL). Lze tedy teoreticky mít pro jednu službu více bran s různými vazbami, tj. volat jednu službu různými způsoby, např. první SOAP přes HTTP, druhou SOAP přes HTTP nad SSL a třetí SOAP přes SMTP.

**Výpis 2-3:** Ukázka WSDL souboru pro webovou službu vracející průměrnou teplotu

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions name="Pocasi" targetNamespace="urn:diplomka:priklad:Pocasi"
      xmlns:tns="urn:x-example:services:StockQuote" xmlns:xsd1="http://example.com/pocasi.xsd"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/" >
<types>
     <schema targetNamespace="http://example.com/pocasi.xsd" xmlns="http://www.w3.org/2000/10/XMLSchema">
          <element name="VratPrumerTeplotu">
                    <complexType>
                              <all>
                                        <element name="misto" type="string" />
                              </all>
                    </complexType>
          </element>
          <element name="VratPrumerTeplotuOdpoved">
                    <complexType>
                              <all>
                                        <element name="prumer-teplota" type="float" />
                              </all>
                    </complexType>
          </element>
     </schema>
</types>
<message name="VratPrumerTeplotuInput">
          <part name="body" element="xsd1:VratPrumerTeplotu" />
</message>
<message name="VratPrumerTeplotuOutput">
          <part name="body" element="xsd1:VratPrumerTeplotuOdpoved" />
</message>
<portType name="PrumerTeplotaPortType">
          <operation name="VratPrumernouTeplotu">
                    <input message="tns:VratPrumerTeplotuInput" />
```

```
                <output message="tns:VratPrumerTeplotuOutput" />
            </operation>
    </portType>
    <binding name="PrumerTeplotaSoapBinding" type="tns:PrumerTeplotaPortType">
            <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
            <operation name="VratPrumernouTeplotu">
                <soap:operation soapAction="" />
                <input>
                    <soap:body use="literal" />
                </input>
                <output>
                    <soap:body use="literal" />
                </output>
            </operation>
    </binding>
    <service name="PrumerTeplotaService">
            <port name="PrumerTeplota" binding="tns:PrumerTeplotaSoapBinding ">
                <soap:address location="http://teplota.pocasi.asmx" />
            </port>
     </service>
</definitionst>
```

***Zdroj:*** *Vlastní*

Vazby odkazují na rozhraní (`portType`), které je souhrnem operací (`operation`). Rozhraní v objektově-orientovaných jazycích odpovídá objektu, jednotlivé operace odpovídají metodám objektu, nebo v jazyce C funkcím.

Každá operace definuje obvykle dvě zprávy (`message`), jednu vstupní a jednu výstupní, ale může i méně. Každá zpráva obsahuje žádnou, jednu nebo více částí (`part`), které odpovídají parametrům a návratovým hodnotám. Z toho plyne, že volané funkce mohou mít více návratových hodnot než jen jednu.

Typy parametrů a návratových hodnot jsou definovány pomocí XML Schema [6]. Jako jednoduché typy jsou tedy k dispozici řetězce (`string`), čísla s pohyblivou (`float`) i pevnou (`decimal`) řádovou čárkou, pravdivostní hodnoty (`boolean`), binární data (`base64Binary`), časový okamžik (`dateTime`), časový interval (`duration`), URL (`anyURI`); dále jejich odvozeniny - výčtové typy,

číselné intervaly, záporná čísla (`negativeInteger`), celá čísla různého rozsahu (`int, byte, short, long`). Je možné definovat složené typy vzniklé jako souhrny nebo varianty z jiných typů, jednoduchých i složených, dokonce lze definovat jeden typ jako rozšíření druhého, lze tedy vyjádřit dědičnost objektů.

K dispozici je specifikace WSDL [5].

## 2.5 UDDI

Jedním z důležitých standardů pro webové služby je UDDI (Universal Description, Discovery and Integration). Každá z těchto dílčích funkcí je zastřešena odpovídajícími standardy, jejichž základem je XML. K nalezení a identifikaci webových služeb nebo obchodních partnerů pro e-business (tedy proces Discovery), slouží UDDI adresář. Za popis (Description) odpovídá standard WSDL a integrace nebo transport dat (Integration) probíhá pomocí protokolu SOAP.

UDDI může odpovědět na následující otázky:

- Jaká rozhraní webových služeb jsou publikována, která jsou založena na WSDL a které je určeno pro určitý obor podnikání, průmysl?
- Jaké společnosti implementovaly své služby postavené na jednom z těchto rozhraní?
- Jaké webové služby, nabízené v určité kategorii, jsou dnes nabízeny?
- Jaké webové služby nabízí určitá společnost?
- Koho musím kontaktovat abych mohl používat webové služby určité společnosti?
- Jaké jsou detaily určité webové služby?

Specifikace UDDI jsou zlatými stránkami webových služeb. Stejně jako u tradičních zlatých stránek můžete hledat firmy nabízející požadované služby, přečíst si základní fakta o nabízených službách a kontaktovat někoho k získání dalších informací. Webovou službu můžete samozřejmě nabízet bez registrace v UDDI, stejně jako když otevřete obchůdek v přízemí vašeho domu a spolehnete se na „ústní“ reklamu mezi sousedy. Pokud však chcete obsáhnout významnější trh, potřebujete UDDI, aby vás zákazníci mohli najít.

Vstupem do adresáře UDDI je soubor XML, který popisuje podnik a jím nabízené služby. Zápis do adresáře UDDI je možno rozdělit na tři části. „Bílé stránky“ popisují firmu nabízející službu: název, adresu, kontakty atd. „Žluté stránky“ zahrnují průmyslové kategorie založené na standardních systematikách, jako je North American Industry Classification System a Standard

Industrial Classification. „Zelené stránky“ popisují dostatečně podrobně rozhraní ke službě, aby kdokoli mohl napsat aplikaci používající Webovou službu. Způsob, jakým jsou služby definovány je uveden v dokumentu UDDI nazvaném Technology Model nebo tModel (dále jen tModel). V mnoha případech obsahuje tModel soubor WSDL, který popisuje rozhraní SOAP k webové službě, ale tModel je dostatečně pružný, aby mohl popsat téměř jakýkoli druh služby.

Adresář UDDI rovněž obsahuje několik způsobů, jak vyhledávat služby potřebné ke stavbě vašich aplikací. Můžete například hledat poskytovatele služby v určeném geografickém místě nebo podnik určitého typu. Adresář UDDI pak dodá informace, kontakty, odkazy a technická data, podle kterých vyhodnotíte, které služby splňují vaše požadavky.

UDDI umožňuje vyhledat podniky, od kterých můžete získat určité webové služby. Co když však již víte, s kým chcete obchodovat, ale nevíte jaké služby nabízí? Specifikace WS-Inspection [16] umožňuje procházet množinu webových služeb nabízených na určitém serveru. Zde můžete zjistit, která služba splňuje vaše požadavky.

### 2.5.1 Popis funkce UDDI registru

UDDI záznamy jsou uchovávány operátory uzlů. Jsou to společnosti, které se zavázali provozovat veřejný uzel, který vyhovuje specifikacím konsorcia UDDI.org [9]. V dnešní době existují 4 veřejné uzly, které vyhovují specifikaci UDDI verze 2. Provozují ho společnosti Microsoft, IBM, Hewlett Packard a SAP.

Operátoři jsou povinni replikovat data mezi sebou skrze zabezpečený kanál. Data mohou být publikována na jednom uzlu a po replikaci mohou být nalezena i na ostatních uzlech. V dnešní době probíhá replikace každých 24 hodin. V budoucnu se očekává zkrácení tohoto intervalu v závislosti na tom, jak bude narůstat počet aplikací závislých na datech z UDDI.

UDDI se chová stejně jako například registr Windows, spoléhá se na Globální Unikátní Identifikátor (GUID), který garantuje možnost vyhledávání a nalezení umístění zdrojů. UDDI dotazy směřují především k vyhledání rozhraní (WSDL soubor, XSD soubor apod.) nebo k umístění webové služby.

UDDI používá informační model, který je založen na XML. UDDI definuje čtyři druhy informací, které uživatel potřebuje k tomu, aby mohl používat webovou službu poskytovatele [8]:

- Business Information (obchodní informace)
- Service Information (informace o službě)
- Binding Information (informace o spojení)
- data o specifikacích služeb (informace o rozhraní)

Obchodní informace se ukládají do tzv. BEE (Business Entity Element). Sem patří např. jméno a adresa poskytovatele služeb. Data Business Entity rovněž umožňují uživateli vyhledat určitou službu - podobně jako v oborovém telefonním seznamu - například takovou, která je k dispozici v určitém regionu nebo se týká určité oblasti. Business Entity Element funguje také jako jakýsi informační kontejner, ve kterém jsou obsaženy všechny ostatní informace o webové službě (Service and Binding Informations) [8].

Informace o službě jsou uloženy v elementu BSE (Business Service Elementu). Ten slouží k rozdělení služeb do skupin, například na základě obchodních procesů, pro které byly koncipovány. Příkladem obchodních procesů jsou například služby e-businessu pro nákup zboží nebo logistické služby. Technická data pro aplikace, jejichž pomocí naváže uživatel kontakt s webovou službou a komunikuje s ní, dodávají Binding Templates. Do této kategorie patří například webová adresa, pod kterou je služba dosažitelná [8]. .

Ještě podrobnější technická data poskytuje jeden podsoubor Binding Templates, element tModel. Ten specifikuje rozhraní, které webová služba používá. TModel je obsahuje jedinečný klíč, který rozhraní webové služby popisuje. Mají-li dvě služby stejný klíč, znamená to, že jsou koncipovány na stejných standardech, a jsou tedy navzájem kompatibilní [8].

Přístup k adresáři probíhá buďto před webové rozhraní nebo programově. Druhý způsob nabízí více možností a uskutečňuje se pomocí API (Application Programming Interface), které se skládá opět ze dvou částí: Inquiry API a Publisher API. Inquiry API je nutné ke čtení dat z UDDI adresáře. Publisher API umožňuje uživateli zadávat informace do adresáře - za předpokladu, že je k tomuto přístupu autorizován. To, jakým způsobem se identifikují firmy nebo oddělení, které chtějí zadávat své nabídky do UDDI adresáře, není předepsáno [8].

Specifikace UDDI [9] je dostupná na webových stránkách společnosti UDDI.org.

## 2.6 Vztahy mezi UDDI a WSDL

WSDL hraje ve webových službách velmi důležitou roli. Je proto důležité pochopit, jak UDDI a WSDL spolu těsně spolupracují a vysvětlit pojem rozhraní versus implementace. Jak WSDL, tak UDDI byly navrženy, aby jasně oddělily abstraktní meta-data a konkrétní implementaci. Porozumění tomuto rozdělení je nezbytné pro pochopení WSDL a UDDI.

Například WSDL rozlišuje zprávy (`messages`) a porty (`ports`): `Messages`, požadovaná syntaxe a sémantika webové služby, jsou vždy abstraktní, zatímco `ports`, síťová adresa, kde bývá webová služba vyvolána, jsou vždy konkrétní. Není tedy nezbytné uvádět informace o portu v souboru WSDL. WSDL soubor může obsahovat pouze abstraktní informace o rozhraní a nemusí poskytovat žádné údaje o implementaci. Takový WSDL soubor, který neobsahuje informace o implementaci, je považován za validní.

Validita WSDL souboru s sebou přináší jednu obrovskou výhodu. Může totiž existovat více implementací jednoho konkrétního WSDL rozhraní. Tento návrh umožňuje neslučitelným systémům implementovat stejné rozhraní a současně zaručit, že tyto systémy můžou spolu navzájem komunikovat. Jestliže tři různé společnosti implementují ten samý WSDL soubor, a bude existovat klientský software, který na základě WSDL souboru vytvoří určitý proxy/stub kód, pak může tento klient komunikovat se všemi třemi implementacemi bez nutnosti měnit kód klientské aplikace. Stačí jen v programu měnit adresy přístupového bodu k webové službě.

UDDI pracuje se stejným modelem oddělení abstrakce od implementace pomocí konceptu zvaného tModel. Například nějaká instituce nebo skupina podniků může publikovat standardní rozhraní pro určité odvětví či obor. Podle tohoto rozhraní pak několik společností může implementovat svoje služby. Každá taková implementace se bude odkazovat na jediný tModel. Příkladem takového tModelu bývá nejčastěji WSDL soubor. Vztah WSDL rozhraní a jeho implementace je analogický vztahu tModel a Binding Template na UDDI.

## 2.7 Standardní rozhraní

Standardy vznikají většinou dvěma různými způsoby: nějaká skupina se rozhodne definovat standard nebo se standardem stane implementace jedné organizace. Organizace které se zabývají vytvářením standardů jako ANSI, ISO a W3C jsou příklady skupin, které určují co jsou standardy.

Druhý způsob vzniku standardů je takový, že existuje implementace, která se postupem času díky svému širokému používání a užívání začne považovat za standard. Takovýto druh standardů se nazývá „de facto standard“. Příkladem takového standardu je například IBM PC. Když IBM navrhla svoji verzi osobního počítače, mnoho společností vyrábějících počítačové komponenty se přizpůsobilo, přestože se oficiální standardizace neuskutečnila.

„De facto standardy“ se objevují v mnoha dalších odvětvích a hrají v oblasti webových služeb důležitou roli. Praktickým příkladem takového standardu je společné a vhodně definované komunikační rozhraní, které může být sdíleno partnery v nějakém oboru či odvětví, což sebou přináší podmínky pro efektivní obchodování.

Je velice nepravděpodobné, aby dvě vytvořené webové služby nabízející stejnou funkcionalitu měly identické rozhraní, které popisuje WSDL dokument (předpokládejme, že jsou vytvářeny izolovaně, tj. bez jakékoli předchozí dohody). To téměř jistě znemožní komunikaci mezi oběma stranami. Pokud bychom chtěli tyto systémy propojit, nevyhneme se úpravám systémů na jedné či druhé nebo obou stranách. To je však časové náročné a přináší to s sebou dodatečné náklady, které mohou snadno převýšit přínosy.

Tyto problémy odstraňuje standardní rozhraní. Největší výhodou standardního rozhraní k webové službě je, že umožňuje aplikaci komunikující přes toto rozhraní, komunikovat s jakoukoli další webovou službou, která společné rozhraní implementuje.

## 2.8 Současný stav v oblasti webových služeb

Technologie webových služeb je poměrně nová a prochází velmi rychlým vývojem. Dosud jsme mluvili o tom, jak hovořit s webovými službami (SOAP), jak jsou webové služby popsány (WSDL) a jak webové služby najít (UDDI). Toto vše vytváří sadu základních specifikací, které jsou základem pro integraci a agregaci aplikací. Z těchto základních specifikací staví podniky reálná řešení a získávají z nich reálnou hodnotu.

I když bylo již vykonáno mnoho práce, aby se webové služby staly realitou, stále ještě zbývá spousta práce. V současné době již lidé úspěšně využívají webové služby, existují však stále věci, které jsou dosud jen cvičením pro vývojáře, například zabezpečení, operační management, transakce, spolehlivá výměna informací (messaging). Architektura globálních webových služeb (GXA) navrhovaná společnostmi Microsoft, IBM, BEA… může posunout webové služby na další

úroveň poskytnutím promyšleného, obecného modelu pro přidání nových pokročilých modulárních a rozšiřitelných funkcí webových služeb.

S tím jak se bude architektura globálních webových služeb vyvíjet, budou uváděny specifikace pro tyto a další potřeby.

## 2.9 Bezpečnost dat

V počátečních fázích vývoje se na SOAP pohlíželo jako na protokol založený na HTTP, takže se předpokládalo, že SOAP bude mít stejnou úroveň zabezpečení jako HTTP. Dnes existují tisíce webových aplikací se zabezpečením na úrovni HTTP, takže toto zabezpečení se zdálo přiměřené i pro SOAP. Současný standard SOAP předpokládá, že zabezpečení je věcí přenosu a dále se zabezpečením nezabývá.

Jakmile se SOAP rozšířil a stal se více obecným protokolem běžícím na vrcholu řady přenosů, stal se problém zabezpečení závažnějším. Protokol SOAP byl navržen jako stavební blok, takže naštěstí se již pracuje na specifikacích vycházejících ze SOAP, které poskytnou přídavné bezpečnostní funkce pro webové služby.

Zabezpečení webových služeb řeší specifikace WS-Security, která je základem Architektury globálních webových služeb. Společnosti Microsoft, IBM a VeriSign předložily WS-Security příslušným standardizačním institucím [10] a očekává se její následná implementace mnoha výrobci. Kombinace modelu, specifikací a standardizačního procesu zabezpečení webových služeb umožní podnikům bez obav vyvíjet bezpečné všestranně využitelné webové služby a rychle a efektivně zvýšit bezpečnost existujících aplikací těchto služeb.

WS-Security definuje standardní sadu rozšíření SOAP, využitelných k implementaci integrity a „utajitelnosti“ (confidentiality) v aplikacích webových služeb. WS-Security popisuje standardní mechanismy k výměně zabezpečených podepsaných zpráv v prostředí webových služeb a poskytuje důležitý základ, který pomůže vývojářům budovat bezpečnější, široce využitelné aplikace webových služeb.

Specifikace WS-Security a celkový plán „Security in a Web Services World“ jsou k dispozici na následujících webových sídlech:

- IBM developerWorks [17]
- Microsoft MSDN [18]
- VeriSign [19]

### 2.9.1 SOAP a zabezpečený přenos dat

Samotný protokol SOAP neřeší otázky zabezpečení. Byl vytvořen k jednoduchému zasílání zpráv mezi počítači. SOAP však nabízí rozšiřitelný rámec, který dovoluje zahrnovat informace o zabezpečení do přenášených zpráv.

Bezpečnost je složité téma, proto si dovolím shrnout základní cíle v oblasti zabezpečení v otevřeném prostředí:

- Integrita zpráv zajišťuje, že zpráva není při přenosu modifikována.
- Utajitelnost zajišťuje, že zpráva může být přečtena pouze zamýšleným příjemcem.
- Důkaz původu dokazuje příjemci, že zpráva opravdu pochází od odesílatele.
- Vzájemné prokázání identity (autentikace) umožňuje klientovi ověřit identitu služby a službě ověřit identitu klienta.
- Autorizace omezuje přístup k službám.

Nejběžnějším mechanismem používaným k zajištění integrity zpráv, utajitelnosti zpráv a autentikaci je zabezpečení na úrovni transportu, jako SSL nebo TLS. SSL/TLS používá k zabezpečení zpráv šifrování pomocí veřejného klíče. Pokud odesílatel zašifruje zprávu použitím jeho soukromého klíče (proti příjemcově veřejnému klíči), zajišťuje SSL/TSL důkaz původu. Vše se odehrává v transportní vrstvě, takže SSL/TSL zabezpečení je naprosto transparentní pro komunikující SOAP aplikace.

Je důležité si uvědomit, že prozatím byla řeč o HTTP zabezpečení a ne o SOAP zabezpečení. SOAP je formát pro přenášení zpráv a není svázán s konkrétním protokolem. Pokud se spolehnete pouze na HTTP zabezpečení, jste omezeni pouze na tento jediný protokol. Přitom SOAP zpráva může být přenášena přes více protokolů.

Běžný může být požadavek, aby zpráva byla šifrována takovým způsobem, že zprávu nemůže číst nikdo kromě odesílatele a konečného příjemce. Mechanismus HTTP přes SSL chrání pouze zprávy při transportu z jednoho síťového portu na druhý. Pokud je použit tento způsob zabezpečení, každý

uzel nebo aplikace nacházející se za cílových portem a kterým SOAP zpráva prochází, má možnost číst vaše data, což nemusí být žádoucí.

Pro jednoduchou komunikaci je SSL/TSL často dostatečné, ale s rostoucími nároky bývají potřeba i další bezpečnostní opatření.

Představme si například situaci, kdy chceme spolu s objednávkou zaslat platební informace. Může to být číslo kreditní karty nebo bankovní příkaz. Budeme pochopitelně požadovat, aby tyto informace nebyly veřejně přístupné. Proto je třeba tyto informace zašifrovat a tu část požadavku, kde se tyto důvěrné informace nacházejí, podepsat digitálním podpisem. Část zprávy je třeba doručit odbytovému středisku pro vyřízení objednávky, část musí být doručena účetnímu středisku kvůli platební transakci a celá zpráva může být například vložena do nějaké evidence.

Odbytové středisko potřebuje znát předmět objednávky, ale nemusí znát platební informace. Tímto způsobem vystavíme důvěrné informace lidem, kteří je nepotřebují znát. V případě, že použijeme SSL mechanismus je zpráva zašifrována pouze při cestě transportním kanálem. Potom již každý uzel či aplikace přes které zpráva prochází, má možnost tyto důvěrné informace číst.

Výhoda SOAP zabezpečení spočívá v tom, že můžeme zašifrovat platební informace, včetně veřejného klíče banky. Druhá strana (dodavatel) může zašifrovaný platební příkaz zaslat bance, kde je rozšifrován a banka může provést peněžní převod. Když banka zašle potvrzení, že byly peníze převedeny, dodavatel může odeslat objednané zboží bez toho, aby se dozvěděla něco o bankovním účtu odběratele.

Pro přenášení zabezpečených zpráv s element `Body` používá k přenosu obsahu šifrované zprávy, zatímco element `Header` se použije k přenosu informací o zabezpečení. Mohou to být autentizační informace (jméno, heslo), digitální podpisy, šifrovací klíče atd.

# 3 Studie nasazení webových služeb

## 3.1 Ukázkové řešení

Cílem mého ukázkového řešení není vytvořit e-business řešení, které je možné nasadit do reálného prostředí, ale demonstrovat návrh a implementaci webových služeb na konkrétních příkladech.

Mějte prosím na vědomí, že vytvářené aplikace (klientská a serverová implementující webové služby) jsou jednoduché a slouží především pro studijní účely. Vytvořený systém neřeší všechny činnosti, které by řešil skutečný informační systém v dodavatelsko-odběratelském řetězci.

V následujících kapitolách vysvětluji teoretické principy vytváření webových služeb, které jsou provázeny mými myšlenkami a úvahami. Snažím se co možná nejvěrohodněji zachytit reálný problém.

Řešení popisuje vývoj webových služeb od samotné myšlenky na jejich zavedení až po samotnou implementaci. To vše je také provázeno praktickými ukázkami komunikace a XML souborů, které tvoří základ řešení.

### 3.1.1 Omezení

Jak jsem již uvedl, je systém značně zjednodušený. Za významné omezení, které stojí za to předem zmínit, je způsob, jakým se identifikují nabízené produkty. Pro všechny produkty se používají jednoznačné identifikátory, s kterými jsou všechny subjekty seznámeny. V reálné situaci tomu tak pravděpodobně nebude a tento problém by se řešil například způsobem, že se v klientské aplikaci zajistí nějaký převodní mechanismus identifikátorů dodavatele na vlastní identifikátory. Například u knih to nemusí být potřeba, protože identifikátorem bude pravděpodobně ISBN.

## 3.2 Popis řešeného problému

Problém, který jsem se rozhodl řešit pomocí webových služeb, jsem vybral z oboru nákupu a prodeje knih, hudby, software a filmu. Myslím, že tento obor činnosti je srozumitelný každému a navíc je velmi snadné určit vlastnosti popisující jednotlivé druhy produktů. [1]

Cílem ukázkového řešení je vytvořit takový systém, který bude zajišťovat komunikaci v dodavatelsko-odběratelském řetězci. Na jedné straně bude figurovat distribuční centrum, které pořizuje zboží od dodavatelů (nakladatelů, vydavatelů) a toto nakoupené zboží poté bude nabízet širokému okruhu svých zákazníků (především velkoobchodům nebo maloobchodům). Na straně druhé bude stát neomezený počet dodavatelů (nakladatelů, importérů …), kteří budou nabízet své zboží. Nabízený sortiment zboží může obsahovat knihy, software, filmy a hudbu.

[1] Například u knih to může být ISBN, autor, počet stran, rok vydání …, u filmu například režisér, herci atd.

## 3.3 Výchozí stav

Kdysi bývalo běžné, že se zboží objednávalo telefonickým způsobem. Tento způsob je však velmi neefektivní, protože je časové náročný a nákladný. Následovalo faxování a elektronická pošta, jehož hlavním problémem bylo, že objednávající neměl přehled o nabízeném sortimentu, jeho cenách a stavu na skladě. Dnes je již celkem běžné, že je možné zadávat objednávky přes webové stránky. Tento způsob odstraňuje již zmíněné nedostatky, objednávající má možnost nalézt celý nabízený sortiment dodavatele, jeho ceny, případně jeho stav na skladě. Stále tu ale přetrvává problém s tím, že každá webová stránka dodavatele je jedinečná. Způsob navigace ve stránkách, vyhledávání produktů a jejich objednávání je u každého dodavatele jiný. Zaměstnanec z oddělení zásobování stále stráví mnoho času procházením webových stránek různých dodavatelů, jejich porovnáváním a dalšími činnostmi až po vytvoření objednávky.

## 3.4 Návrh

Ideálním řešením je jednoduchá aplikace, která získává informace o různých dodavatelích a umožňuje vyhledávání nabízeného sortimentu založené na vlastnostech produktu, ceně, dodacích podmínkách, platebních podmínkách, stavu na skladě apod. a nakonec zadání objednávky.

A právě vytvoření takového systému je možné s pomocí webových služeb. Webové služby jdou však mnohem dále a umožňují sdílet informace a automatizovat procesy v tomto dodavatelsko-odběratelském řetězci. Například odběratel (distributor) může zadávání objednávek automatizovat takovým způsobem, že pokud se množství zboží na prodejním skladě sníží pod určitou úroveň je možné zaslat příslušnému dodavateli objednávku na chybějící zboží, která se provede automaticky, to znamená bez přičinění některého ze zaměstnanců. Takové řešení pak umožňuje dosáhnout vysoké efektivity, šetří čas i náklady.

### 3.4.1 Požadavky

Budeme chtít vytvořit webovou službu, která zajistí komunikaci mezi distributorem a dodavateli knih, hudby, filmu a softwaru (nakladateli, importéry). Veškerou aktivitu v tomto řešení ponese distributor, který od zavedení informačního systému postaveného na webových službách očekává největší přínos.

Požadavkem distributora je mít k dispozici vždy optimální skladbu sortimentu. Nezbytným požadavkem jsou krátké dodací lhůty. Pro velké odebírané množství pak distributor dosahuje výhodných podmínek, především cenových a dodacích. Tento sortiment distributor nabízí širokému okruhu svých zákazníků, především velkoobchodům a maloobchodům za přijatelné ceny.

Můžeme si ovšem také představit, že iniciativa tohoto řešení může vycházet od nakladatele nebo importéra, který si bude chtít zajistit co možná největší odbyt tím, že se pokusí získat co nejvíce odběratelů, distributorů nebo přímo maloobchodníků.

Distributor bude získávat informace o nabízených produktech dodavatelů (nakladatelů knih, vydavatelů hudby...) a z nich pak bude vytvářet svoji vlastní nabídku pro své koncové zákazníky. Prakticky to bude znamenat, že distributor bude volat webové služby všech dodavatelů či importérů, kteří vyvinou jisté úsilí na vytvoření webových služeb a budou ochotni spolupracovat.

Odběratel (distributor) tak získá možnost pořizovat zboží od dodavatelů a dodavatel získá odbytiště pro své zboží se všemi výhodami elektronické komunikace. Můžeme si představit, že aplikace distributora je schopna díky technologii UDDI automaticky vyhledat všechny dodavatele požadovaného zboží a zadat objednávku. To vše je pomocí webových služeb technicky možné bez zásahu člověka. Něco takového zatím v praxi běžné není, protože zvláště obchodní procesy často vyžadují jisté vyjednávání o kontraktu apod. především kvůli rizikům plynoucích z obchodního styku. Nic ale nebrání automatizaci nákupních transakcí mezi důvěryhodnými obchodními partnery. Se zavedením potřebné legislativy a se zlepšením možností provádět zabezpečené platby přes internet se tato situace změní.

### 3.4.2 Jednotlivé kroky

Distributor vytvoří standardní rozhraní pro komunikaci ve svém oboru a toto rozhraní zveřejní. Dále vyvine klientskou aplikaci, která je schopna komunikovat s webovými službami implementující toto rozhraní. Ostatní dodavatelé budou moci na základě zveřejněného rozhraní implementovat své webové služby, což umožní propojit jejich systém se systémem distributora a automatizovat obchodní procesy.

Nový dodavatel se rozhodne zefektivnit svůj odbyt a rozhodne se pro elektronickou komunikaci. Pokud ještě odběratele, který umožňuje komunikovat přes webové služby nezná, může ho nalézt například na UDDI. S tímto odběratelem (distributorem) se zkontaktuje a dohodne se na vzájemné

spolupráci. Na základě popisu webových služeb získaných na UDDI vytvoří či pouze upraví svůj informační systém tak, aby komunikace prostřednictvím webových služeb mohla být realizována.

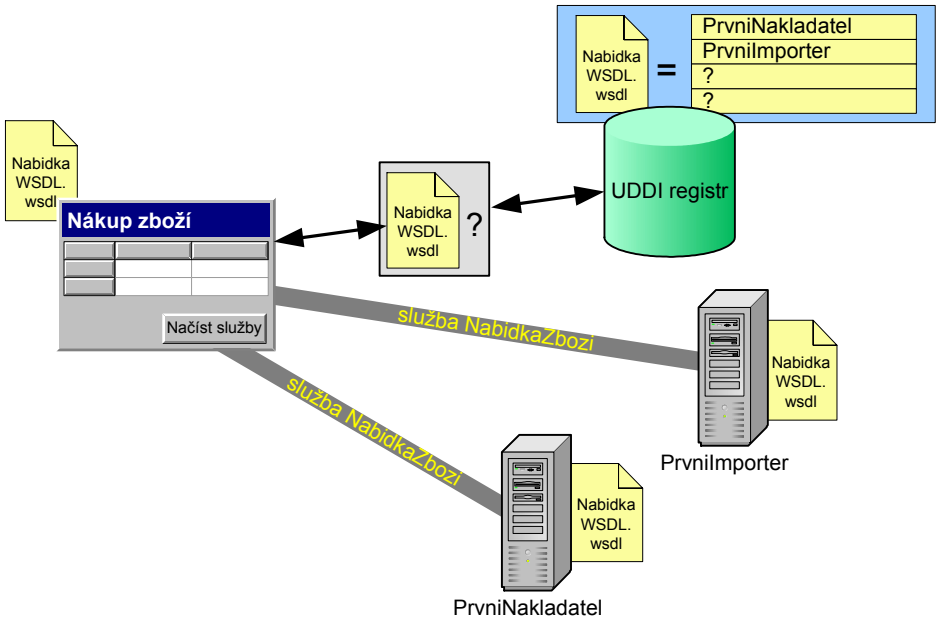
Někdo si může položit otázku, co dělat v případě, že dodavatel již poskytuje webové služby, které poskytují rozhraní, které je nekompatibilní s rozhraním distributora. V zásadě se nabízí 3 základní možnosti.

1) zpřístupnit webové služby přes další rozhraní (tj. rozhraní distributora)
2) vlastní transformace - jelikož SOAP požadavek/odpověď je vlastně XML dokument, lze využít například jazyka XSLT (Extensible Stylesheet Language Transformations), pomocí kterého je možné celou obálku či její část přetransformovat do požadované podoby
3) specializovaný software - produkty jako například Biztalk [20] dokáží poměrně snadno transformovat příchozí dokumenty na vlastní

Náklady na zavedení mohou snadno převýšit přínosy. Záleží pochopitelně na konkrétní situaci, ale podle svých zkušeností si myslím, že varianta první bude ve většině případů nejefektivnější.

### 3.4.3 Aplikační model

**Obr. 3-1:** Aplikační model




***Zdroj:*** *Vlastní*

# 4 Implementace systému

## 4.1 Obecný postup

1. Vytvoření společného rozhraní.
2. Registrace rozhraní na UDDI serveru.
3. Vytvoření webové služby která společné rozhraní implementuje.
4. Registrace webové služby na UDDI serveru.
5. Vytvoření klientské aplikace.

## 4.2 Vytvoření společného rozhraní

### 4.2.1 Úvaha před vytvářením rozhraní

Ještě než začneme vytvářet webové služby, je velmi vhodné zjistit, zdali již neexistují nějaké standardy, které se hodí pro naše záměry. Zejména to jsou rozhraní (WSDL dokumenty) nebo datové typy (schémata). Pro nalezení těchto informací můžeme využít UDDI registr nebo se můžeme obrátit na obchodní partnery případně konkurenty v našem oboru.

Pokud žádné standardy neexistují, rozhodněte se, zdali není užitečné vytvořit takové webové služby, které již jiná společnost v našem oboru poskytuje.[2] Pokud existující webové služby splňují požadavky vašich zákazníků, je velmi pravděpodobné, že budou vhodné i pro ostatní konkurenty v našem oboru. V tom případě můžeme iniciovat návrh na zavedení standardního způsobu komunikace tj. využívání společného rozhraní. Pokud se na tomto řešení s ostatními konkurenty dohodneme, přinese to užitek nejen nám jako poskytovateli webových služeb, ale i konkurenci.

Pro účely této diplomové práce předpokládejme, že žádné vhodné rozhraní neexistuje. Distributor tedy bude vytvářet zcela nové rozhraní, které bude následně registrováno na UDDI.

[2] Ve skutečnosti to znamená použít pouze existující rozhraní a datové typy (WSDL dokument), vlastní implementace bude jistě rozdílná a není pro vzájemnou komunikaci důležitá.

Jak již bylo popisováno v předchozí části této diplomové práce, aby klient mohl webovou službu používat, musí dopředu znát určité informace, konkrétně pak znalost rozhraní. Podle tohoto rozhraní je vytvořena klientská aplikace, která může komunikovat s určitou webovou službou, která toto rozhraní implementuje. Výhodou je, že klientská aplikace může komunikovat s neomezeným množstvím webových služeb. Jedinou podmínkou je společného rozhraní. Tento způsob umožňuje klientské aplikaci komunikovat i s novými webovými službami, které v době vytvoření klientské aplikace ještě neexistovaly.

Toto rozhraní včetně používaných datových typů je popsáno v dokumentu WSDL. Následující část je zaměřena na zásady, které je vhodné při vytváření dokumentu WSDL dodržet a vše je demonstrováno na konkrétním příkladě z ukázkové aplikace.

### 4.2.2 Zásady při vytváření rozhraní

Navrhněte takové rozhraní, které vyhovuje potřebám vašich klientů. Soustřeďte se jak budou vypadat přenášená data po síti, ne jak vytvořit webovou službu pomocí nějakého programovacího prostředku.

Většina programovacích nástrojů totiž dokáže kompletní WSDL automaticky vygenerovat na základě napsaného kódu. Nástroje pro automatické generování WSDL podle napsaného kódu stále prochází vývojem a mají určité nedostatky.

Různé platformy nemusí mít identické datové typy a další konstrukce, proto se může stát, že vygenerované schéma v prostředí .NET Framework od Microsoftu nemusí být identické s vygenerovaných schématem pomocí aplikace dodávané například s platformou J2EE od Sunu nebo BORLAND C++BUILDER 6 od Borlandu. Vygenerovaný popis rozhraní (WSDL soubor) je navíc úzce svázán s konkrétní implementací webové služby. To by nebyl problém, pokud rozhraní webové služby bude mít pouze jednu implementaci. Ale cílem je vytvořit takové rozhraní (standardní), které bude implementovat více subjektů (dodavatelů).

Je vhodné začít s psaním WSDL rozhraní dřív, než s vlastním programováním tříd a následném generování WSDL dokumentu. Klienti na různých platformách si pak na základě WSDL dokumentu potřebné třídy vytvoří. Tím, že budete myslet na rozhraní místo na implementaci,

získáte maximální výhody plynoucí z interoperability namísto toho, co vyhovuje vašemu programovacímu prostředí.

Neméně důležitým požadavkem je vytvoření takového rozhraní, které upřednostňuje zasílání většího množství dat v jedné zprávě místo většího množství zpráv k získání požadovaných dat. Například požadavek na získání objednávky by měl vrátit také informace o zákazníkovi, protože nám to může ušetřit další volání. Hlavním důvodem je skutečnost, že zprávy přenášené po síti mají určité zpoždění.

Existuje množství aplikací na vytváření WSDL souborů. Použil jsem produkt Cape Clear 4 od společnosti CAPE CLEAR SOFTWARE [15], který je velmi dobře hodnocený a je především zdarma.

### 4.2.3 Návrh rozhraní v dodavatelsko-odběratelském řetězci

Rozhodl jsem se rozdělit webovou službu na dvě části. Důvod je ten, že na službu pro objednávání zboží jsou zavedeny určité kontrolní mechanismy, aby služba nebyla dostupná každému, protože by mohlo dojít k jejímu zneužití. Služba pro získávání informací o nabízeném sortimentu je dostupná všem.

Důsledkem této skutečnosti bude existence dvou WSDL dokumentů.

WSDL dokument může být logicky rozdělen na 2 části:

- Abstraktní definice - Types, Message, Operation, Port Type
- Konkrétní definice - Binding, Port, Service

Jak již bylo v předchozím textu uvedeno, aby mohlo být rozhraní standardizováno, musí WSDL dokument obsahovat pouze takové definice, které nejsou spjaty s konkrétní implementací webové služby. Z tohoto důvodu nebude vytvořený WSDL dokument obsahovat definice pro elementy `Port` a `Service`.

Zbývající definice jsou rozděleny do nezávislých dokumentů. XML schéma s definicemi datových typů, které se používají v obou webových službách, tvoří jeden soubor. Definice formátu zpráv pro obě služby jsou umístěny také v samostatném souboru. Tato technika umožňuje znovupoužití některých definic a usnadňuje údržbu WSDL dokumentů.

K oddělení jednotlivých definic se používá element `import`, který umožňuje oddělení různých elementů v definici webové služby do nezávislých dokumentů, které jsou importovány podle potřeby.

**Tab. 4-1:** Rozdělení WSDL dokumentů

| Název metody | Popis |
|---|---|
| DataTypes.xsd | XML schéma s definicemi datových typů používaných v obou webových službách |
| NabidkaMessageTypes.xsd | XML schéma s definicemi formátu zpráv používaných ve webové službě pro vyhledávání sortimentu |
| ObjednavaniMessageTypes.xsd | XML schéma s definicemi formátu zpráv používaných ve webové službě pro objednávání zboží |
| NabidkaWSDL.wsdl | WSDL dokument popisující rozhraní webové služby pro vyhledávání sortimentu |
| ObjednavaniWSDL.wsdl | WSDL dokument popisující rozhraní webové služby pro objednávání zboží |

***Zdroj:*** *Vlastní*

**Tab. 4-2:** Přehled webových služeb

| Název | Programový název | Popis |
|---|---|---|
| Nabídka zboží | NabidkaZbozi | slouží k vyhledávání a zjišťování informací o nabízeném sortimentu. Dále pak k získání nabízených kategorií a zboží |
| Objednávání zboží | ObjednavaniZbozi | slouží především k objednávání zboží |

***Zdroj:*** *Vlastní*

**Tab. 4-3:** Přehled metod, které bude webová služba Nabídka zboží implementovat.

| Název metody | Popis |
|---|---|
| VratNabizeneProdukty | vrací sortiment všech nabízených produktů všech druhů |
| VratPripravovaneProdukty | vrací připravované produkty, které v nejbližší době rozšíří nabízený sortiment |
| VratSeznamPolozek | vrací základní informace o všech nabízených položkách, které odpovídají požadovanému typu produktu |
| VratDetailPolozky | vrací podrobné informace o nabízené položce, které odpovídá identifikátor a typ produktu |
| VratRecenze | vrací recenze, které se vztahují k požadovanému typu produktu |
| VratRecenzeProduktu | vrací recenze na konkrétní položku |
| HledejProdukty | vrací pole položek jednoho typu produktu, které vyhovují zadaným kriteriím |
| DodavatelInfo | vrací informace o dodavateli |

***Zdroj:*** *Vlastní*

**Tab. 4-4:** Přehled metod, které bude webová služba Objednávání zboží implementovat.

| Název metody | Popis |
|---|---|
| ZadatObjednavku | slouží k vytvoření nové objednávky |
| VratObjednavku | vrací požadovanou objednávku |
| VratSeznamObjednavek | vrací seznam objednávek zákazníka |
| VratSkladoveStavy | vrací skladové stavy nabízeného sortimentu |
| ZaplatitObjednaneZbozi | slouží k zaplacení objednaného zboží |

***Zdroj:*** *Vlastní*

### 4.2.4 Vytvoření společného rozhraní

Vytvořil jsem nejdříve rozhraní webové služby v programovém kódu. Následně jsem vygeneroval odpovídající WSDL soubor. V předchozím textu bylo uvedeno, že jsou s tímto postupem spojeny určité problémy. Jak za chvíli budete mít možnost vidět, vyhnul jsem se tomuto problému jistou kličkou.

Vygenerované soubory (pro obě webové služby) jsem ručně v editoru WSDL rozdělil na více částí, abych oddělil definice týkající se konkrétní implementace. Provedl jsem ještě další úpravy spojené s tím, aby byly jednotlivé soubory tvořily jeden celek. Navíc bylo nutné odstranit velké množství zbytečných informací, které tam generátor WSDL přidává.

Získal jsem tímto způsobem vygenerované soubory tvořící WSDL dokument, který je oddělen od konkrétní implementace, což bylo zamýšleno.[3]

Adresy vytvořených souborů WSDL jsou v tabulce (Tab. 4-5).

### 4.2.5 Abstraktní definice

Při vytváření WSDL dokumentu je vhodné začít s vytváření abstraktních definic.

#### *4.2.5.1 Datové typy (element types)*

K definici datových struktur používaných ve zprávách lze použít teoreticky libovolný typový systém, ale nejčastěji se používají XML schémata [6]. Nástroje pro webové služby se starají o mapování datových typů podle XML schémat na nativní datové typy použitého jazyka.

Jak bylo uvedeno výše, v našem fiktivním řetězci se budou vyskytovat čtyři druhy zboží. Jsou to knihy, hudba, filmy a software. Všechny používané datové typy ukládám do samostatného souboru s příponou .xsd, která se používá pro XML schéma.

Pro zefektivnění nakládání s těmito prvky jsem použil dědičnost, takže každý prvek dědí ze společné základové třídy *Polozka*. Tato třída obsahuje několik vlastností, které jsou společné pro všechny druhy zboží. Je to identifikátor, název zboží, doporučená cena, naše cena, jazyková verze, vydavatel, popis zboží a klasifikace, která obsahuje seznam kategorií, kam je konkrétní produkt

[3] Tento způsob neobhajuji, použil jsem ho pro zjednodušení práce.

zařazen. Jednotlivé odvozené třídy pak mají přiřazeny své specifické vlastnosti. Pro knihu je to například autor, ISBN, rok vydání, počet stran a vazba.

XML schéma s definicí třídy *Polozka* a odvozené třídy *Kniha* (kompletní XML schéma je k dispozici v příloze [Příloha B] vypadá následovně.

**Výpis 4-1:** Ukázka z XML schématu (soubor DataTypes.xsd)

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="CommonDataTypes" xmlns:tns="urn:diplomka-zahradka:types" elementFormDefault="qualified"
           targetNamespace="urn:diplomka- zahradka:types" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="Polozka" abstract="true">
        <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="1" name="Nazev" type="xs:string" />
                <xs:element minOccurs="0" maxOccurs="1" name="DoporucenaCena" type="xs:double" />
                <xs:element minOccurs="1" maxOccurs="1" name="NaseCena" type="xs:double" />
                <xs:element minOccurs="1" maxOccurs="1" name="Jazyk" type="tns:JazykVerze" />
                <xs:element minOccurs="0" maxOccurs="1" name="Popis" type="xs:string" />
                <xs:element minOccurs="1" maxOccurs="1" name="Vydavatel" type="xs:string" />
                <xs:element minOccurs="0" maxOccurs="1" name="Klasifikace" type="tns:Klasifikace" />
        </xs:sequence>
        <xs:attribute name="Id" type="xs:int" />
    </xs:complexType>
    <xs:complexType name="Kniha">
        <xs:complexContent mixed="false">
            <xs:extension base="tns:Polozka">
                <xs:sequence>
                        <xs:element minOccurs="1" maxOccurs="1" name="Autor" type="xs:string" />
                        <xs:element minOccurs="1" maxOccurs="1" name="Isbn" type="xs:string" />
                        <xs:element minOccurs="1" maxOccurs="1" name="RokVydani" type="xs:int" />
                        <xs:element minOccurs="0" maxOccurs="1" name="PocetStran" type="xs:int" />
                        <xs:element minOccurs="0" maxOccurs="1" name="Vazba" type="tns:TypVazby" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    . . .
</xs:schema>
```

***Zdroj:*** *Vlastní*

Všechny používané datové typy jsou definovány ve jmenném prostoru urn:diplomka-zahradka:types. K jejich definici se přitom používají standardní datové typy ze specifikace W3C [6].

#### 4.2.5.2 Zprávy (element message)

Dále je vhodné pokračovat s definicí formátu předávaných zpráv pomocí dříve definovaných datových typů, které budou protokolem SOAP přenášeny. Zprávy fungují jako vstupní anebo výstupní struktury pro operace.

Formát zprávy je nezávislý na tom, zda se používá přístup přes HTTP, SMTP nebo jiný protokol. Při použití webových služeb musíme rozlišit dva typy zpráv - požadavky a odpovědi. Požadavky jsou zprávy, které posílá klient službě, odpovědi přichází ze služby na klienta.

Narozdíl od předchozího XML schématu, budou zprávy již specifické pro každou část webové služby. Službě pro získávání informací o nabízeném sortimentu přiděluji jmenný prostor urn:diplomka-zahradka:NabidkaWsdl, službě pro objednávání zboží urn:diplomka-zahradka:ObjednavaniWsdl.

Ukázka ze schématu s formátem přenášených zpráv pro službu, která umožňuje získat informace o nabízeném sortimentu, vypadá následovně.

**Výpis 4-2:** Ukázka z XML schématu (soubor NabidkaMessageTypes.xsd)

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="NabidkaTypes" xmlns:tns="urn:diplomka-zahradka:NabidkaWsdl"
        targetNamespace="urn:diplomka-zahradka:NabidkaWsdl" xmlns:q1="urn:diplomka-zahradka:types"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import namespace="urn:diplomka-zahradka:types" />
    <xs:element name="VratSeznamPolozek">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="1" maxOccurs="1" name="TypProduktu" type="q1:ProduktTyp" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="VratSeznamPolozekResponse">
        <xs:complexType>
```

```
            <xs:sequence>
              <xs:element minOccurs="0" maxOccurs="1" name="VratSeznamPolozekResult" type="q1:ArrayOfPolozka" />
            </xs:sequence>
          </xs:complexType>
      </xs:element>
      . . .
</xs:schema>
```

***Zdroj:*** *Vlastní*

V předchozí ukázce je zobrazena definice formátu pro zprávu *VratSeznamPolozek.* Tato zpráva, stejně jako většinou používaných zpráv, je rozdělena na dvě části. První část (element `VratSeznamPolozek`)[4] definuje formát pro odchozí zprávu (požadavek). Druhá část (element `VratSeznamPolozekResponse`) definuje formát pro příchozí zprávy (odpovědi).
Požadavek přijímá jeden parametr a na základě jeho hodnoty vrací odpověď. Tato zpráva, kterou zasílá klient službě se skládá z elementu `TypProduktu`. Tento element typu `ProduktTyp` je definován v XML schématu a je z vlastního jmenného prostoru.

Odpověď webové služby obsahuje element, který je typu pole *Polozek,* které je také definováno v XML schématu.

Následuje ukázka definice přenášené zprávy.

**Výpis 4-3:** Ukázka z WSDL dokumentu (soubor NabidkaWSDL.wsdl)

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions xmlns:i1="urn:diplomka-zahradka:types" xmlns:i0="urn:diplomka-zahradka:NabidkaWsdl"
          targetNamespace="urn:diplomka-zahradka:NabidkaWsdl" . . .>
      . . .
      <message name="VratSeznamPolozekIn">
          <part name="parameters" element="i0:VratSeznamPolozek" />
      </message>
       <message name="VratSeznamPolozekOut">
          <part name="parameters" element="i0:VratSeznamPolozekResponse" />
       </message>
```

[4] Název formát pro příchozí zprávu se může jmenovat například VratSeznamPolozekRequest, ale rozhodl jsem se dodržet konvenci, kde formát odchozích zpráv pojmenovávám stejně jako název zprávy a formátu odchozích zpráv přidávám koncovku Response.

```
    ...
</definitions>
```

***Zdroj:*** *Vlastní*

Pro popis struktury zpráv ze používá element `message`, který obsahuje nula nebo více elementů `part`. Element `part` odpovídá parametru nebo návratové hodnotě vzdálené procedury.

Pro metodu *VratSeznamPolozek* (ostatní metody budou vyjmenovány v další kapitole (viz. 4.2.5.3) jsou definovány dvě zprávy. Zpráva s názvem *VratSeznamPolozekIn* je definice požadavku a *VratSeznamPolozekOut* definuje odpověď.

Zpráva *VratSeznamPolozekIn* je typu *VratSeznamPolozek*, jehož formát je definován v XML schématu pro přenášené zprávy (viz. Výpis 4-2). Analogické je to i pro zprávu s názvem *VratSeznamPolozekOut*.

Specifikace WSDL nespecifikuje konvence pro pojmenovávání zpráv. Pokud používáme nějaký nástroj pro generování WSDL, je pravděpodobné, že tento nástroj bude mít vlastní konvenci pro pojmenovávání zpráv. V mém případě přidávám do názvu zprávy směr (IN jako input pro vstupní zprávu, OUT jako output pro výstupní zprávu).

### 4.2.5.3 Operace (element operation)

Element `operation` odpovídá ve WSDL specifikaci metodě. U operace se definuje jaké má vstupy a výstupy. Vstup a výstup je popsán již existující zprávou (viz. Výpis 4-3). Definice operací jsou sdruženy do elementu `portType`.

**Výpis 4-4:** Ukázka z WSDL dokumentu (soubor NabidkaWSDL.wsdl)

```
<portType name="JZ2004WSNabidka">
    <operation name="VratRecenzeProduktu">
        <documentation>Metoda vraci vsechny recenze na jeden pozadovany produkt</documentation>
        <input message="i0:VratRecenzeProduktuSoapIn" />
        <output message="i0:VratRecenzeProduktuSoapOut" />
    </operation>
    ...
</portType>
```

***Zdroj:*** *Vlastní*

V předchozí ukázce můžeme vidět definici operace *VratRecenzeProduktu*. Kromě nepovinného popisu obsahuje operace vstupní a výstupní zprávu, která je definována ve stejném souboru (viz. Výpis 4-3).

### 4.2.6 Konkrétní definice

Nyní provedeme převod z abstraktních datových typů, zpráv a operací do konkrétní fyzické reprezentace zpráv.

#### *4.2.6.1 Vazba (element binding)*

Pro definování konkrétních aspektů operací se používá element `binding`. Slouží pro navázání určitého rozhraní `portType` na konkrétní protokol a formát přenosu zpráv.

**Výpis 4-5:** Ukázka z WSDL dokumentu (soubor NabidkaWSDL.wsdl)

```
<binding name="JZ2004WSNabidka" type="i0:JZ2004WSNabidka">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
. . .
<operation name="VratRecenze">
    <soap:operation soapAction="urn:diplomka-zahradka/VratRecenze" style="document" />
        <input>
            <soap:body use="literal" />
        </input>
        <output>
            <soap:body use="literal" />
        </output>
   </operation>
. . .
</binding>
```

***Zdroj:*** *Vlastní*

Souhrnu operací, které se vztahují k rozhraní (portType) pro vyhledávání nabízeného sortimentu (viz. Výpis 4-4). přiřazujeme protokol pro posílání zpráv (SOAP) a přenosový protokol (HTTP). Tyto informace vyjadřuje použití elementu `binding` s prefixem „soap“ spolu s hodnotou atributu transport.

V ukázce můžeme vidět, že pro operaci *VratRecenze*, která byla dříve definována, nastavujeme hodnotu „SOAPAction“. SOAPAction je HTTP hlavička, kterou klient posílá, když vyvolává službu. Ta je použita k nalezení a vyvolání odpovídající metody.

#### *4.2.6.2 Brána (element port) a služba (element service)*

Tyto elementy popisují již konkrétní implementaci webové služby. Z tohoto důvodu nejsou do popisu rozhraní přidány. Zabývat se jimi má smysl v kapitole Vytváření webových služeb, ale já se jimi záměrně nezabývám, protože definice těchto elementů se vytváří dynamicky při spuštění webové služby. To automaticky zajišťuje programové prostředí ve kterém jsou webové služby provozovány. Přístup k takovému dokumentu je možný, pokud se za URL přístupového bodu služby přidá parametr WSDL.

Například požadavek http://jznet.aerohosting.cz/nabidka.asmx?WSDL vrátí následující informace (viz. Výpis 4-6)

**Výpis 4-6:** WSDL dokument vrácený webovou službou

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:s=http://www.w3.org/2001/XMLSchema
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:i0="urn:diplomka-zahradka:NabidkaWsdl"
        xmlns:tns="urn:diplomka-zahradka" targetNamespace="urn:diplomka-zahradka"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
    <import namespace="urn:diplomka-zahradka:NabidkaWsdl"
          location="http://www.volny.cz/jiri_z/Diplomka/WSDL3/NabidkaWSDL.wsdl" />
   <types />
   <service name="NabidkaZbozi">
          <documentation>Služba pro zjištění nabízeného sortimentu knih, hudby, filmu a software</documentation>
          <port name="JZ2004WSNabidka" binding="i0:JZ2004WSNabidka">
              <soap:address location="http://jznet.aerohosting.cz/nabidka.asmx" />
          </port>
   </service>
</definitions>
```

***Zdroj:*** *Vlastní*

Z předchozího výpisu zjistíme, že název webové služby je NabidkaZbozi. Element `port` definuje bránu, jejíž vazba je JZ2004WSNabidka (definovaná výše viz. Výpis 4-6) a její přístupový bod je

http://jznet.aerohosting.cz/nabidka.asmx. Důležitou informaci obsahuje element `binding`, který odkazuje na soubor WSDL, kde je definováno standardní rozhraní pro tuto webovou službu.

## 4.3 Registrace rozhraní na UDDI serveru

K registraci rozhraní zvolíme testovací UDDI server od společnosti Microsoft http://test.uddi.microsoft.com. Je to v podstatě replika produktivního UDDI serveru [21], která je určená pro testovací účely. Rozdíl spočívá v tom, že záznamy z tohoto testovacího UDDI serveru se nereplikují na UDDI servery ostatních operátorů.

Publikování na UDDI je velmi přímočarý proces. Před vlastním procesem registrace je třeba rozmyslit následující body.

1. Určit tModels (dokumenty WSDL), které bude webová služba implementovat. Pokud rozhraní není na UDDI registrováno a proto tModel neexistuje, je třeba vytvořit nový, který nové rozhraní reprezentuje.
2. Připravit si název a základní informace o společnosti, případně nějaký kontakt na odpovědné osoby.
3. Zvolit jaké kategorie a identifikace nejlépe reprezentuje vaší společnost. Na výběr je množství taxonomií, mezi nima například North American Industry Classification System (NAICS), Universal Standard Products and Services Codes (UNSPSC), ISO 3166, Standard Industry Classification (SIC), GeoWeb Geographic Classification a další.
4. Rozhodnout se, jaké služby bude vaše společnost nabízet.
5. Zařadit webové služby do příslušné kategorie. Stejně jako mohou být kategorizovány společnosti, mohou být i jednotlivé služby.

### 4.3.1 Publikování na UDDI

V tento moment máme na výběr ze dvou možností. Můžeme se registrovat pomocí webového rozhraní nebo programově. Pro programový přístup byla vytvořena specifikace UDDI Version API Specification [23], kterou ve svých programovacích nástrojích implementovaly vývojářské firmy (například Microsoft nabízí balík Microsoft UDDI SDK).

Volíme registraci pomocí webového rozhraní testovacího UDDI uzlu [22]. Je to velmi intuitivní proces, který začíná registrací. V tomto případě, tj. publikace v testovacím prostředí se neklade

důraz na popis společnosti, stačí jen emailová adresa a heslo. Po přihlášení již máme možnost vytvářet záznamy o naší společnosti (Business Information), webových službách (Service Information), spojení (Binding Information) nebo tModelech.

Protože registrujeme rozhraní, zajímá nás především nabídka tModel. Vytvoříme dva nové tModely, které zastupují rozhraní (WSDL dokumenty). Vyplníme požadované záznamy pro oba tModely. Nejdůležitější informacemi jsou adresa WSDL dokumentu a zařazení do kategorií, které umožňuje snadné vyhledávání.

V následující tabulce jsou informace o umístění souborů, které tvoří WSDL dokumenty.

**Tab. 4-5:** Adresy WSDL dokumentů

| Soubor | URL |
|---|---|
| DataTypes.xsd | http://www.volny.cz/jiri_z/Diplomka/WSDL/ |
| NabidkaMessageTypes.xsd | http://www.volny.cz/jiri_z/Diplomka/WSDL/ |
| ObjednavaniMessageTypes.xsd | http://www.volny.cz/jiri_z/Diplomka/WSDL/ |
| NabidkaWSDL.wsdl | http://www.volny.cz/jiri_z/Diplomka/WSDL/ |
| ObjednavaniWSDL.wsdl | http://www.volny.cz/jiri_z/Diplomka/WSDL/ |

***Zdroj:*** *Vlastní*

V nabídce poskytovatelé (Providers) se vyplňují informace o společnosti. Protože distributor žádné služby nenabízí, vyplníme jen informace o společnosti, kontaktní informace a zařadíme společnost do kategorií v některé z vhodných taxonomií.

Webová rozhraní jednotlivých UDDI operátorů se mohou mírně lišit. Data jsou však interně uložena ve formátu XML. Následuje ukázka definice tModelu.

**Výpis 4-7:** TModel pro rozhraní NabidkaRozhrani

```
<TModel TModelKey="uuid:ce342b3b-3bda-4788-8bc3-ec2b687354f3" authorizedName="Jiri Zahradka">
  <name>Diplomka:NabidkaRozhrani</name>
  <description>
    Rozhraní pro získávání nabízeného sortimentu
  </description>
  <overviewDoc>
    <overviewURL>
```

```
        http://www.volny.cz/jiri_z/Diplomka/WSDL/NabidkaWSDL.wsdl
      </overviewURL>
    </overviewDoc>
    <categoryBag>
        <keyedReference TModelKey="uuid:70a80f61-77bc-4821-a5e2-2a406acc35dd"
                name="ntis-gov:sic:1987" value="5300" />
        <keyedReference TModelKey="uuid:c0b9fe13-179f-413d-8a5b-5004db8e5bb2" name="ntis-gov:naics:1997"
                value="421" />
        <keyedReference TModelKey="uuid:4e49a8d6-d5a2-4fc2-93a0-0411d8d19e88"
                 name="uddi-org:iso-ch:3166:1999" value="CZ-LI" />
    </categoryBag>
. . .
</TModel>
```

***Zdroj:*** *Vlastní*

Element `categoryBag` je nepovinný a obsahuje kategorie, do kterých je rozhraní zařazeno.

Obrázky obrazovek webového rozhraní pro registraci na UDDI jsou k dispozici v příloze [Příloha B].

Na závěr této kapitoly bych rád upřesnil důležitou skutečnost. Fiktivní společnost, nazval jsem ji HlavniDistributor, vytváří pouze rozhraní webových služeb. Tento distributor očekává, že toto rozhraní budou implementovat jeho dodavatelé a tím se vytvoří dodavatelsko-odběratelský řetězec, které bude propojeno webovými službami.

Můžeme si představit situaci, kdy jeden z dodavatelů (například nakladatel) vytvoří rozhraní a webovou službu, která toto rozhraní implementuje. Ačkoli je to jiný případ, pro distributora a ostatní dodavatele se vůbec nic nemění. Důležité je, že rozhraní existuje a na jeho základě vytváří distributor klientskou aplikaci a ostatní dodavatelé implementují své webové služby.

## 4.4 Vytvoření XML webové služby

Webová služba sloužící k vyhledávání nabízeného sortimentu (NabidkaZbozi) nebude nijak zabezpečená a bude dostupná všem, tj. nejen nám jako distributorovi, ale i dalším potenciálním klientům (odběratelům).

V případě služby pro objednávání (ObjednavaniZbozi) je zaveden mechanismus, který nebude umožňovat objednávat zboží každému. Tato webová služba bude zabezpečena tak, aby přístup k ní měli jen registrovaní klienti. V prvé řadě to bude distributor, který byl nazván HlavniDistributor.

### 4.4.1 Postup

Narozdíl od předchozí kapitoly, kde rozhraní vytvářel distributor, webovou službu vytváří každý z dodavatelů, který má zájem se na obchodní spolupráci podílet.

K vytvoření webové služby potřebujeme popis rozhraní neboli WSDL dokument. Ten můžeme získat přímo, například od tvůrce rozhraní (v tomto případě je to HlavniDistributor) nebo můžeme využít UDDI registr. Ve skutečnosti není důležité jak ho získáme.

Ve stručnosti popíši způsob jak může při vyhledávání rozhraní pomoci UDDI registr, což je také jedna z jeho nejdůležitějších funkcí. Na domovské stránce UDDI registru [22] zvolíme nabídku Search, která nám umožní vyhledávat podle stanovených kritérií.

Informace o požadovaném WSDL dokumentu můžeme získat vyhledáváním podle kategorií, poskytovatelů nebo přímo vyhledáním tModelu. Tak například v nabídce vyhledávání podle tModelu zadáme jako vyhledávácí kritérium řetězec %Nabidka% a aplikace mi vrátí požadovaný tModel, jehož celý název je Diplomka:Nabidka. Při zobrazení podrobností o tomto tModelu získám adresu umístění WSDL dokumentu.

Pokud máme k dispozici WSDL dokument, můžeme použít některý ze SOAP nástrojů, který umožňuje komunikaci s webovou službou přes její rozhraní. V mém případě jsem použil vývojové prostředí Microsoft .NET Framework SDK.

Na základě WSDL dokumentu vytvoříme serverovou třídu, která bude použita k vytvoření webové služby, která definované rozhraní implementuje. Kromě serverové třídy, která rozhraní implementuje, se vygenerují i další pomocné třídy, především pro používané datové typy (viz. [Příloha B]).

Hlavním úkolem implementátora webové služby je napsat kód pro funkce, které budou vykonávány, když budou přicházet požadavky přes rozhraní naší webové služby. Seznam všech implementovaných funkcí je totožný se seznamem operací ve WSDL dokumentu, protože právě na

základě WSDL dokumentu jsou generovány serverové třídy. Touto částí se víc zabývat nebudu, protože psaní programového kódu nespadá do tématu mé diplomové práce.

Webová služba umístěná na webovém serveru čeká na požadavky klientů a v okamžiku, kdy přes HTTP přijde zpráva SOAP, spustí webovou službu a předá jí požadavek. Data požadavku jsou deserializována z XML a předána jako parametry příslušné metodě. Výsledek služby je pak zpětně serializován do XML a předán zpět klientovi jako odpověď. Následuje ukázka SOAP požadavku a odpovědi.

**Výpis 4-8:** SOAP požadavek

```
POST /NakladatelWS/NabidkaZbozi.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: xxx
SOAPAction: "urn:diplomka-zahradka/VratSeznamPolozek"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <VratSeznamPolozek xmlns="urn:diplomka-zahradka:NabidkaWsdl">
      <TypProduktu>Kniha</TypProduktu>
    </VratSeznamPolozek>
  </soap:Body>
</soap:Envelope>
```

***Zdroj:*** *Vlastní*

**Výpis 4-9:** SOAP odpověď

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <VratSeznamPolozekResponse xmlns="urn:diplomka-zahradka:NabidkaWsdl">
      <VratSeznamPolozekResult>
        <Polozka Id="50001">
```

```
            <Nazev xmlns="urn:diplomka-zahradka:types">Zapisovatelé otcovský lásky</Nazev>
              <NaseCena xmlns="urn:diplomka-zahradka:types">240</NaseCena>
              <Jazyk xmlns="urn:diplomka-zahradka:types">CZ</Jazyk>
              <Vydavatel xmlns="urn:diplomka-zahradka:types">Nakladatelství  Petrov</Vydavatel>
              <Klasifikace xmlns="urn:diplomka-zahradka:types">
                 <Kategorie Id="11" Nazev="Krásná" />
              </Klasifikace>
              <Autor xmlns="urn:diplomka-zahradka:types">>Michal Viewegh</Autor>
              <Isbn xmlns="urn:diplomka-zahradka:types">>80-7227-022-2</Isbn>
              <RokVydani xmlns="urn:diplomka-zahradka:types">1998</RokVydani>
              . . .
          </Polozka>
          <Polozka>
            . . .
          </Polozka>
          . . .
      </VratSeznamPolozekResult>
    </VratSeznamPolozekResponse>
  </soap:Body>
</soap:Envelope>
```

***Zdroj:*** *Vlastní*

V ukázce (viz. Výpis 4-8) je zobrazena SOAP obálka pro volání metody *VratSeznamPolozek* s předávaným parametrem *TypProduktu*, který má hodnotu Kniha.

SOAP odpověd (Výpis 4-9) obsahuje element `VratSeznamPolozekResult`, který obsahuje pole elementů `Polozka`.

Z WSDL dokumentu si můžeme ověřit, že obsah elementu `Body` v požadavku i v odpovědi přesně odpovídá definicím.

## 4.5 Registrace XML služby na UDDI serveru

Když máme vytvořeny obě webové služby, které implementují rozhraní pro vyhledávání nabízeného sortimentu a objednávání zboží, můžeme umístit funkční verzi na webový server.

Abychom umožnili její nalezení, zaregistrujeme tyto služby v UDDI registru. Publikujeme především tři zásadní informace:

- Informace o společnosti
- Umístění webové služby
- Rozhraní, které služba implementuje (tModel)

Nejdříve je nutné si uvědomit, zdali je WSDL rozhraní již registrováno. Při registraci webové služby je nutné zadat odkaz na použitý tModel. Z tohoto důvodu je třeba registrovat tModel ještě před samotnou webovou službou. V tomto případě víme, že WSDL rozhraní je již na UDDI registrováno distributorem.

V první řadě musí dodavatel publikovat záznamy o své společnosti. Vymyslíme pro tento účel fiktivního dodavatele, nazval jsem ho PrvniNakladatel. Vytvoříme nový záznam, kde vyplníme požadované údaje o společnosti, kontaktní informace včetně zařazení do kategorií v některé z vhodných taxonomií.

Pokud tak učiníme, můžeme přidat novou webovou službu do nabídky webových služeb. Nové webové službě je nutné přidat jméno, popis, dále je vhodné zařadit ji do příslušných kategorií v některých z nabízených taxonomií stejným způsobem jako při klasifikaci společnosti.

Důležitou informací je element `BindingTemplate`. Tento element reprezentuje implementaci webové služby, kam patří především protokol a formát přenášených zpráv a síťová adresa. Každá webová služba může mít několik elementů `BindingTemplate`, například může být vytvořena pro komunikaci kromě protokolu HTTP například SMTP a další.

V neposlední řadě je nutné zadat adresu, kde bude webová služba respektive její implementace (`BindingTemplate`) přístupná. Tím je tzv. endpoint (přístupový bod), který může obsahovat URL, emailovou adresu nebo jinou formu adresy.

Zbývá svázat element `BindingTemplate` s odpovídajícím tModelem. K tomu potřebujeme znát klíč tModelu (tModelKey). Pokud ho neznáme, webové rozhraní nám umožní získat tuto hodnotu z názvu tModelu, který se pro službu NabidkaZbozi jmenuje Diplomka:NabidkaRozhrani.

Analogií tohoto postupu publikujeme i druhou službu ObjednavaniZbozi, jejíž implementaci svážeme s tModelem Diplomka:ObjednavaniRozhrani.

Následující ukázka zobrazuje informace o webové službě ve formátu XML, což je interní formát pro uložení UDDI informací. Přístup k těmto informacím zpravidla zajišťuje uživatelské webové rozhraní.

**Výpis 4-10:** UDDI záznam pro webovou službu

```
<businessService serviceKey="uuid:4ba2f0f9-335f-4086-a8de-d675443e7439"
            businessKey="uuid:3c274720-3489-4e51-ba31-85e0af6015f9">
 <name>NabidkaZbozi</name>
 <description>Služba pro získávání informací o nabízeném_sortimentu knih, hudby, filmů a software.</description>
     <bindingTemplates>
       ...
    </bindingtemplates>
    <categoryBag>
     …
    </categoryBag>
</businessService>
<bindingTemplate serviceKey="uuid:4ba2f0f9-335f-4086-a8de-d675443e7439"
            bindingKey="uuid: eef229ba-e808-4ea9-808c-85d26a4d91d9">
   <description>NabidkaZbozi SOAP binding.</description>
   <accessPoint URLType="http">http://jznet.aerohosting.cz/nabidka.asmx</accesspoint>
   <TModelInstanceDetails>
       <TModelInstanceInfo TModelKey="uuid:ce342b3b-3bda-4788-8bc3-ec2b687354f3">
         <instanceDetails>
             <overviewDoc>
                <description>Odkazuje na standardní WSDL rozhraní webové služby.</description>
                <overviewURL>http://www.volny.cz/jiri_z/Diplomka/WSDL/NabidkaWSDL.wsdl</overviewURL>
             </overviewDoc>
         </instanceDetails>
       </TModelInstanceInfo>
   </TModelInstanceDetails>
</bindingTemplate>
```

***Zdroj:*** *Vlastní*

**Tab. 4-6:** Dostupné implementace webových služeb

| Poskytovatel | Název služby | Adresa |
|---|---|---|
| PrvniNakladatel | NabidkaZbozi | http://jznet.aerohosting.cz/Nakladatel/nabidka.asmx |
| | ObjednavaniZbozi | http://jznet.aerohosting.cz/Nakladatel/objednavani.asmx |
| PrvniImporter | NabidkaZbozi | http://jznet.aerohosting.cz/Importer/sortiment.asmx |
| | ObjednavaniZbozi | http://jznet.aerohosting.cz/Importer/objednavka.asmx |

***Zdroj:*** *Vlastní*

Předchozí tabulka (viz. Tab. 4-6) zobrazuje dostupné webové služby, které existovaly v době psaní této práce. V této práci se nezmiňuji o implementaci webových služeb dalšího dodavatele (PrvniImporter). Webové služby tohoto dodavatele i postup registrace na UDDI jsou zcela identické. Je přidán pro rozšíření aplikační infrastruktury a věrnější zachycení problému (více dodavatelů je běžné).

Registrací webové služby na UDDI je umožněno ostatním klientům tyto služby dodavatelů nalézt a volat jejich metody .[5]

## 4.6 Vytvoření klientské aplikace

Shrňme si, co bylo doposud provedeno. Máme vytvořený WSDL dokument popisující rozhraní pro dvou webových služeb (vyhledávání nabízeného sortimentu a objednávání). Tato rozhraní byly registrovány na UDDI fiktivním distributorem (HlavniDistributor). Na základě těchto WSDL definic rozhraní byly fiktivními dodavatelu (PrvniNakladatel, PrvniImporter) implementovány dvě webové služby. Tyto webové služby byly registrovány na UDDI.

Cílem je vytvořit takovou aplikaci, která bude přes společné rozhraní komunikovat s webovými službami jiných společností (dodavatelů). Umožní získávat informace z UDDI registru, především seznam webových služeb implementující společné rozhraní. Jakmile získá přístupové body (endpoints) ke službám, je vytvořen základ pro propojení klientské aplikace s aplikací dodavatele.

[5] V ukázkovém modelu je jediným klientem distributor. Avšak není problém, aby se další klienty do modelu začlenil.

Hlavní body při vytváření klientské aplikace jsou:

- Vytvoření proxy třídy
- Vytvoření programového kódu, který vyhledává přístupové body implementující požadované rozhraní
- Použití nalezených přístupových bodů v aplikaci

### 4.6.1 Vytvoření proxy třídy

Klientem webových služeb je fiktivní HlavniDistributor. V tomto případě je to právě HlavniDistributor, který na sebe převzal iniciativu a vytvořil standardní rozhraní.

První úkol, který musí klientský vývojář učinit, je vytvořit proxy třídu. Proxy třídu si můžeme představit jako třídu, která zprostředkovává veškerou komunikaci mezi klientem a serverem (webovou službou). Jejím úkolem je zjednodušit přístup ke službě na takové úrovni, abychom byli zcela oproštěni od úkolu vytvoření dotazu (ve formátu SOAP dokumentu), odeslání dotazu přes zvolený protokol, přijetí a zpracování odpovědi (ve formátu SOAP). Pro vývojáře se práce s webovou službou jeví tak, jako by šlo o třídu umístěnou na lokálním počítači.

#### *4.6.1.1 Funkce proxy třídy*

Klient dynamicky založí objekt proxy a vyvolá jeho metodu. Metoda má stejné jméno, parametry i návratovou hodnotu jako metoda implementovaná na straně webové služby. Proxy serializuje parametry do souboru XML a vytvoří SOAP dokument. Ten pošle požadavek webovému serveru přes protokol HTTP. Odpověď je potom přijata objektem proxy, který vrácená data deserializuje a předává zpět volajícímu kódu klientské aplikace.

K tomu se obvykle používá utilita, která je dodávána s každým vývojovým prostředím či nástrojem pro vytváření webových služeb. Vstupem pro vygenerování takové třídy (tříd) je WSDL dokument.[6]

Vygenerovaná proxy třída je dostupná v příloze [Příloha B].

[6] Některá vývojová prostředí mohou tento proces značně zjednodušit tím, že vytvářejí proxy třídy automaticky na základě odkazu na WSDL dokument.

### 4.6.2 Vyhledání přístupových bodů

V tomto bodě má klientský projekt vytvořeny proxy třídy, ale nemá vůbec žádné ponětí o tom, kde se webové služby fyzicky nacházejí. Zbývá nalézt přístupové body na kterých je webová služba dostupná.
Cílem je získat seznam společností, se kterými může klient přes společné rozhraní komunikovat. Důležité jsou především tyto tři informace:

- Název společnosti
- Koncový bod pro službu Nabídka
- Koncový bod pro službu Objednávání

Kromě výše uvedených informací můžeme získat mnohé další informace. K dotazování UDDI registru byla vytvořena specifikace UDDI Version 2.0 API [23]. V tomto případě se dotazování uskutečňuje pomocí programového prostředku Microsoft UDDI SDK, který výše zmíněnou specifikaci implementuje.

Klientská aplikace zasílá požadavek na UDDI registr s cílem vrátit seznam webových služeb implementujících požadované rozhraní. Parametrem požadavku je klíč tModelu.

### 4.6.3 Použití nalezených přístupových bodů

Vrácený seznam webových služeb implementujících oba tModely spolu s dalšími informacemi může klientská aplikace uchovat ve vhodné datové struktuře.

Pokud při běhu aplikace dojde k požadavku na vyvolání určité metody požadované webové služby, prohledá aplikace seznam získaných webových služeb, ve kterém nalezne příslušný přístupový bod. Přístupový bod požadované služby potom nastaví proxy objektu a ten následně zašle požadavek odpovídající webové službě.

## 4.7 Popis klientské aplikace

Vytvořená klientská aplikace je velmi jednoduchá, plní především následující funkce:

- Nalezení a zobrazení všech dodavatelů implementující webové služby
- Získání nabízeného sortimentu včetně informací o jednotlivých položkách

- Získání recenzí na jednotlivé položky
- Zjištění stavu na skladě u požadovaných položek
- Provedení objednávky
- Získání a zobrazení předešlých objednávek

Několik náhledů obrazovek z této aplikace je k dispozici v příloze [Příloha B]. Zdrojové soubory a zkompilovaná verze je dostupná na CD, které je přílohou této práce [Příloha B].

## 4.8 Zabezpečení

V předchozí části bylo popsáno vytvoření ukázkového modelu založeného na webových službách. Otázka zabezpečení nebyla zmiňována. Tato kapitola se zabývá touto otázkou a popisuje řešení zabezpečení webové služby pro objednávání zboží (ObjednavaniZbozi).

### 4.8.1 Zavedení autentikace

Jak bylo v předchozím textu uvedeno, otázky zabezpečení řeší specifikace WS-Security [17][18][19]. Existuje již několik implementací této specifikace od různých vývojářským společností například WSE 2.0 (Web services Enhacements) od Microsoftu, který jsem pro zabezpečení webové služby použit.

Nejrozšířenější způsob autentikace je zaslání uživatelského jména a hesla. Zahájíme teoretickou úvahou nad tímto způsobem zabezpečení, které nás dovede k řešení, které je ve webové službě použito.

Většina z vás jistě pochopí, že zasílat heslo v nijak nezměněné textové podobě nemusí být ideální. Proto je vhodné zasílat místo původního hesla jeho hash. Dobrá zpráva je, že žádný zlomyslný uživatel nezjistí původní heslo. Může však zachycené hashované heslo odeslat ve svém požadavku, čímž bude autentikován jako originální odesílatel. Abychom eliminovali tento problém, je přidána ochrana navíc.

Kromě hashovaného hesla, zasíláme navíc digest, který obsahuje kombinaci hashovaného hesla, elementu `Nonce` (identifikátor požadavku) a čas vytvoření. V této etapě bude mít každý legitimní požadavek různý hash.

Stále však existuje nebezpečí zneužití, pokud zlomyslný uživatel získá celý uživatelský token (`UsernameToken`) z legitimního požadavku a vloží ho do své zprávy.

Toto riziko se minimalizuje tím, že se požadavku nastaví doba vypršení, která obsahuje kratší časovou hodnotu. K další eliminaci rizika je vhodné, aby si webová služba uchovávala hodnotu elementu `Nonce` a pokud obdržíme více požadavků se stejnou hodnotou, je doporučeno oba požadavky zahodit, protože jeden z nich musí být nelegitimní.

V následující ukázce můžeme vidět, jak vypadá SOAP požadavek implementující základní autentikaci ve webové službě pro objednávání zboží. Pro přehlednost jsou vynechány určité informace popisující způsob zabezpečení, které obsahuje element `Header`.

**Výpis 4-11:** Zabezpečený SOAP požadavek

```
POST /NakladatelWS/NabidkaZbozi.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: xxx
SOAPAction: "urn:diplomka-zahradka/VratSeznamPolozek"

<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/12/utility"
               xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">
  <soap:Header>
      ...
      <wsse:Security soap:mustUnderstand="1">
        <wsse:UsernameToken xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
                    wsu:Id="SecurityToken-c7ef4231-4397-4b4b-ab8d-0ea3fad1f79e">
          <wsse:Username>JiriZ</wsse:Username>
          <wsse:Password Type="wsse:PasswordDigest">NoTe4iN5E=</wsse:Password>
          <wsse:Nonce>QLSVRt9g3e19jJXJYhtBKA==</wsse:Nonce>
          <wsu:Created>2003-12-20T22:37:52Z</wsu:Created>
        </wsse:UsernameToken>
      </wsse:Security>
  </soap:Header>
  <soap:Body>
    <VratSeznamObjednavek xmlns="urn:diplomka-zahradka:ObjednavaniWsdl">
      <IdZakaznika>1001</IdZakaznika>
```

```
      </VratSeznamObjednavek>
   </soap:Body>
</soap:Envelope>
```

***Zdroj:*** *Vlastní*

Podstatné informace obsahuje element `UsernameToken`. Obsahuje element pro uživatelské jméno (`Username`), datum vytvoření požadavku (`Created`), náhodné číslo, které slouží jako identifikátor (`Nonce`) a hashované heslo (`Password`). Atribut Id elementu `UsernameToken` obsahuje digest z hodnot výše uvedených elementů.

Stále tu existuje jedna varianta, kterou předchozí opatření neřeší. Může se stát, že někdo požadavek zachytí, získá z něho uživatelský token, který vloží do svého požadavku a původní legitimní zprávu zničí.

Pro maximální zabezpečení existují dvě možnosti:

- Zavedení SSL/TSL zabezpečení
- Podepsání zprávy SOAP digitálním podpisem

Implementace SSL/TSL zabezpečení nevyžaduje úpravy v aplikaci, je to záležitost čistě administrátorská a proto se jí v této diplomové práci dále nezabývám.

Na druhé straně zavedení digitálního podepisování zpráv již vyžaduje drobné zásahy do programového kódu. Tyto změny však nemají vliv na rozhraní webových služeb a tak jejich zavedení nenaruší jejich funkci.

Výše zmíněné způsoby zabezpečení v ukázkovém řešení nejsou zavedeny. Zavedení SSL zabezpečení je placená služba a zavedení digitálního podepisování zpráv je z programátorského hlediska principiálně stejné jako použití základní autentikace.

## 4.9 Komplexnější scénář ukázkového řešení

### 4.9.1 Propojení s bankou - platební transakce

Přínosem v řešení, které propojuje odběratele s dodavateli, by dozajista bylo zavedení platebních transakcí. V současnosti je v České republice nemožné provádět platby pomocí webových služeb. Tato možnost je uvedena jen pro demonstraci potenciálu webových služeb spolu s elektronickým podpisem.

Pokud odběratel zadává objednávku, musí za objednané zboží pochopitelně zaplatit. Zasílá tedy platební příkaz, který může být součástí objednávky. Ten musí přinejmenším obsahovat identifikaci banky, číslo účtu, částku a datum splatnosti. Pokud jsou tyto informace zasílány nezabezpečeným způsobem, riskujeme jejich odhalení. Pokud takový příkaz banka obdrží, nemůže být zaručeno, že platební příkaz zaslal skutečný majitel účtu.

Řešení spočívá v zašifrování platebních informací a jejich digitálním podepsáním. Kromě toho může být zašifrována a podepsána celá zpráva, pokud obsahuje objednávku. Příjemce zprávy (dodavatel) potom část zprávy se zašifrovanými platebními informacemi, odešle bankovní instituci odběratele. Banka samozřejmě nepřevede peníze ihned, takže žádost nějaký čas čeká. Když je vyřízena, banka zašle notifikaci o převodu peněz zpět dodavateli. Ten může následně vyskladnit zboží.

### 4.9.2 Propojení se systémem dopravce

Integrace informačního systému distributora se systémem dopravce pomocí webových služeb by umožnilo automaticky sledovat informace o dodávce. Tyto informace mohou být například místo, kde se právě zásilka nachází, čas předpokládaného doručení zásilky apod.

# 5 Webové služby a e-business

## 5.1 Informace kdekoli, kdykoli, jakkoli

Vzájemné propojení softwarových aplikací, zařízení a lidí vytváří pozoruhodnou příležitost. Webové služby umožňují jednou vytvořeným aplikacím fungovat na různých zařízení.

Příkladem využití webových služeb můžou být například kalendářové služby, které jsou spojené s mapovými službami, takže vaše auto vám udá směr k místo vašeho dalšího jednání, aniž byste mu to museli přikazovat – může se dokonce spojit s dopravními službami, a pomůže vám tak vyhnout se místům, kde se opravuje silnice nebo kde došlo k dopravní zácpě.

Uvádím další příklady využití webových služeb, z nichž některé patří spíše do blízké budoucnosti, ale čtenář si tak může vytvořit představu, co všechno webové služby dokáží.

Představte si cestovní služby, které komunikují s leteckou společností a změní vám přípojný let, když zjistí, že na místo přestupu nedorazíte včas. Což může být jen první ze série dalších elektronických zásahů, které také změní vaši hotelovou rezervaci nebo rezervaci vypůjčeného automobilu [11].

Představte si služby státní správy, které při libovolném schvalovacím procesu samy komunikují se všemi zainteresovanými úřady a samy vás vyzvou k tomu, co je všechno ještě třeba udělat, a umožní vám to provést z libovolného místa [11].

Mnohé z těchto otevřených webových služeb nebudou přicházet na váš počítač PC, ale na váš osobní komunikátor. Může jím být mobilní telefon, pager, PDA (osobní digitální asistent).

Jednoduše řečeno, každá služba, kterou potřebujete, vám bude poskytnuta právě v tomto okamžiku, kdy ji budete potřebovat - a právě takovým způsobem, jaký je pro vás vhodný a jaký se vám líbí.

Je snadné si představit stovky dalších aplikací, které by mohly být vytvořeny k programování internetu.

## 5.2 Etapy zavedení

Téměř všechny významné společnosti na trhu IS/IT pokládají webové služby za nevyhnutelnost, jediné, v čem se liší, je jejich správné načasování. Technologicky jsou už mnohé současné aplikace a vývojová prostředí na využití webových služeb připravené (Microsoft, Borland, IBM a další) [23].

Podle firmy Sun [13] se rozdělí zavedení webových služeb na tři fáze [23]:

- **Fáze 1** je využití webových služeb v rámci podniku pro integraci mezi různorodými systémy (hardware i software). To by se mělo odehrát v příštích 18 měsících.[7]
- **Fáze 2** znamená implementaci webových služeb pro integraci mezi podniky - partner proti partnerovi - management zásobování, management CRM atd., ta má začít za 18 měsíců.
- Ve **fázi 3** bychom se nakonec asi za tři roky měli dočkat volně vázaných, bezpečných systémů pro koncové uživatele s úplně integrovanými dostupnými bezpečnostními modely, zvláště proto, že tyto normy ještě nejsou k dispozici.

## 5.3 Využití v e-business aplikacích

V současné době je obecně přijat názor, že pro plné využití potenciálu webových služeb je zapotřebí rozsáhlý konceptuální model (framework). Na řešení toto problému jsou zaměřeny mnohé aktivity včetně tvorby GXA navrhované Microsoftem a IBM. Je zcela jasné, že aby webové služby zajistily spolehlivou platformu pro e-business aplikace, musejí vyspět do rozsáhlého a úplného souboru standardů a technologií a v tomto smyslu se stát obdobou procesu vývoje jiných technologií distribuovaného zpracování jako komponentový model J2EE.

Bez ohledu na širokou podporu ze stran dodavatelů IT i IS komunity obecně se v současné době webové služby zatím nevyužívají v širokém měřítku v e-business aplikacích. Podobně jako ostatní technologie využívané v business e-aplikacích i webové služby musejí zajistit podporu základních služeb včetně bezpečnosti, transakcí a tvorby (kompozice) komplexních webových služeb před tím, než bude tato technologie připravena pro využití v e-business aplikacích širokého měřítka. Řada zmíněných standardů je stále v procesu vývoje, a tudíž je spolehlivá platforma webových služeb pro podporu komplexních e-business aplikací stále ještě vzdálena. S ohledem na rychlý rozvoj v této oblasti však nelze pochybovat o tom, že tato technická omezení budou dříve či později vyřešena. Je však otázkou, zda to zajistí automatizaci a skutečnou interoperabilitu e-business aplikací [7].

Za pozitivní lze považovat skutečnost, že kombinace relativně nízkých vstupních nákladů a širokého přijetí standardů webových služeb povede k tomu, že se tento přístup patrně stane preferovanou volbou pro implementaci e-business aplikací [7].

[7] Článek byl zveřejněn 09/2003 [23]

## 5.4 Přínosy pro podnik

Integrace aplikací pomocí webových služeb poskytuje podniku příležitost zvýšit zisk, snížit provozní náklady, šetřit drahocenný čas a snadnější komunikaci mezi zákazníky, zaměstnanci a obchodními partnery.

- **Nižší náklady do IT -** Webové služby využívají otevřené standardy. Lze je snadno vytvářet téměř na všech platformách, proto není třeba dalších investic do podnikového hardware a software. Nezávislost na programovacím jazyce umožňuje vývojářům dále používat jejich preferovaný programovací jazyk a využít jejich existujících znalostí a dovedností. Kromě snížení mzdových a materiálních nákladů je důležitým aspektem také podstatně kratší doba vývoje oproti běžným způsobům integrace podnikových systémů.

- **Zvýšení tržeb -** Zveřejnění obchodních procesů, které nebyly doposud široce dostupné, vede k nalezení nových zákazníků a zvýšení odbytu.

- **Integrace uvnitř podnikových systémů –** Dosud nekompatibilní aplikace v podniku si mohou nyní vyměňovat informace ve formátu XML mezi sebou.

- **Dokonalejší integrace s partnery -** Snazší integrace služeb a aplikací mezi obchodními partnery například v dodavatelském řetězci vede k získání konkurenční výhody.

- **Dokonalejší integrace se zákazníky -** Snazší integrace služeb a aplikací se zákazníky umožňuje podniku nabízet kvalitnější služby a tím získat nové zákazníky a udržet si existující.

- **Rozšíření možností zaměstnanců -** Webové služby umožňují zaměstnancům doručit odpovídající informace v čase a místě kde je právě potřebují. Nezáleží na tom, kde se právě nacházejí, zdali v kanceláři, v automobilu či v divadle, informace dostanou ve vhodné formě na mobilní telefon, PDA, PC či jiné inteligentní zařízení. Naopak informace pocházející od zaměstnance mohou být doručeny ostatním pracovníkům podniku. Nezáleží přitom kde se zaměstnanec právě nachází, když má k dispozici nějaký druh inteligentního zařízení.

# 6 Závěr

I když bylo již vykonáno mnoho práce, aby se webové služby staly realitou především v integraci e-business aplikací, stále ještě zbývá spousta práce. V současné době již lidé úspěšně využívají webové služby a mnoho společností s úspěchem webové služby poskytuje. Na druhé straně však stále existují věci, které jsou dosud jen cvičením pro vývojáře, například zabezpečení, operační management, transakce nebo spolehlivá výměna informací.

Pokud sečteme všechny směry současného ekonomického vývoje, jako integrace systémů uvnitř organizace a mezi obchodními partnery, integrace se zákazníky, možnost využívat různé zdroje dat a nutnost tato data dále zpracovávat, přechod od jednoúčelových systémů typu účetnictví k systémům určeným pro komplexní řízení firmy a přidáme k tomu širokou podporu kolem XML mezi významnými společnostmi na trhu IS/IT a to, že v současné době nemají konkurenční technologii, zjistíme, že webové služby mají před sebou velkou budoucnost.

Hlavním cílem této práce je vysvětlit potenciál webových služeb a jejich využití demonstrovat na konkrétním řešení. Pozornost je věnována teoretickým základům, analýze, návrhu řešení až po jeho implementaci. Výsledkem je jednoduchý systém, který umožňuje komunikaci v dodavatelsko-odběratelském řetězci. V době psaní této práce existují dvě implementace webových služeb, které představují informační systém dvou dodavatelů. Tyto implementace jsou publikovány na UDDI a jsou veřejně dostupné. Klientská aplikace odběratele může s těmito službami komunikovat a umožní odběrateli pořizovat nabízené zboží.

Tento model založený na webových službách je otevřený a díky standardní komunikaci protokolem SOAP umožňuje dalším dodavatelům nebo odběratelům integrovat svoje aplikace s již existujícími a zvyšovat tak počet účastníků tohoto obchodní řetězce.

Hlavní přínos této práce nespočívá ani tak ve vytvoření jednotlivých aplikací, které poskytují nebo konzumují webové služby, jako spíše v návrhu architektury distribuovaného systému a vytvoření základní aplikační infrastruktury, která dokáže propojit aplikace zúčastněných subjektů.

# Slovníček pojmů

**COM** (Component Object Model) **-** původně obecný komponentový model, který ovšem byl rozšířen a posléze i vylepšen Microsoftem na platformě Windows.

**CORBA** (Common Object Request Broker Architecture) **-** poměrně stará obecně navržená multiplatformní architektura nezávislá na programovacím jazyku pro vzdálené volání a předávání objektů, podporovaná různými dodavateli (např. Borland) a použitá jako základ technologie Enterprise JavaBean v Javě.

**DCE** (Distributed Computing Environment) – průmyslový standard, obsahuje z několika. technologií pro distribuovanou komunikaci

**DCOM** (Distributed COM) - distribuovaná verze COM určená pro komunikaci.

**de facto standard -** standard, který přirozeně existuje díky svému širokému přijetí a užívání.

**Digest -** hash vytvořený z textové zprávy. Digest má pevnou délku, zatímco textová zpráva může mít různou a je zpravidla delší. Digest plní funkci jakéhosi zabezpečeného výtahu.

**Disco** (SOAP Discovery) - dokument HTML nebo XML popisující webové služby, které nabízí daný server.

**EDI** (Electronic data interchange) - technologie dovolující uživatelům přenášet mezi jednotlivými počítači sítě informace typu objednávky a faktury.

**Hash -** kontrolní součet, který musí obě strany vypočítat, aby mohla být zpráva ověřena. Používá k ověření, zda obsah zprávy nebyl během přenosu změněn.

**HTTP** (HyperText Transfer Protocol) - internetový protokol sloužící k výměně dat mezi serverem a klientem.

**IIOP** (Internet Inter-Orb Protocol) **-** síťový protokol (jako je třeba FTP nebo HTTP) určený pro vzdálené volání objektů, původně navržený jako přenosový protokol pro CORBA.

**J2EE** (Java 2 Platform Enterprise Edition) - sada technologií distribuovaného zpracování vícevrstvých podnikových aplikací. Základem je programovací jazyk Java.

**MQ Series -** komunikační middleware založený na principu doručování zpráv mezi širokým spektrem softwarových a hardwarových platforem.

**RMI** (Remote Method Invocation)- volání vzdálených objektů v Javě, nejprve řešeno přes proprietární protokol a poté přechod na IIOP.

**RPC** (Remote procedure call) - implementace distribuovaného výpočetního modelu klient/server. Požadavek je zaslán na vzdálený systém, kde je vykonán a výsledek je vrácen klientovi.

**SMTP** (Simple Mail Transfer Protocol) **-** síťový protokol navržený pro přenos emailových zpráv mezi servery a při odesílání emailů z klienta na server.

**TCP/IP** (Transmission Control Protocol/Internet Protoco) - sada protokolů pro internetové sítě. Obsahuje TCP jako primární transportní protokol a IP jako protokol síťové vrstvy.

**Token** - reprezentuje oprávnění k přístupu, nejčastěji obsahuje uživatelské jméno a heslo.

# Seznam literatury a zdroje

[1] Brian E. Travis: XML a SOAP Programování serverů BizTalk. Computer Press; Praha 2000; ISBN 80-7226-303-X.

[2] Jiří Košek:XML pro každého, podrobný průvodce; Grada; Praha 2000; ISBN 80-7169860-1.

[3] Dalibor Kačmář: Programujeme .NET aplikace; Computer Press; Praha 2001; ISBN: 80-7226-569-5.

[4] W3C; SOAP Version 1.2 Part 1; URL: http://www.w3.org/TR/SOAP.

[5] W3C; Web Services Description Language (WSDL) 1.1; URL: http://www.w3.org/TR/wsdl.

[6] W3C; XML Schema Part 2: Datatypes; URL: http://www.w3.org/TR/xmlschema-2/

[7] Elektronické podnikání s využitím webových služeb; URL: http://www.computerworld.cz.

[8] Bernd Reder; UDDI: Žluté stránky pro webové služby; URL: http://www.computerworld.cz.

[9] UDDI.org; UDDI Technical White Paper; URL: http://www.uddi.org.

[10] OASIS; URL: http://oasis-open.org.

[11] SUN: News 5/2001; Otevřené, inteligentní webové služby

[12] SoapWare.org; Implementations; URL: http://www.soapware.org/directory/4/implementations.

[13] SUN Microsystems; URL: www.sun.com.

[14] Worldwide Web Consortium; URL: http://www.w3c.org.

[15] Capeclear; URL: http://www.capeclear.com.

[16] IBM; WS-Inspection Specification; URL: http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html.

[17] IBM; Specification: Web Services Security (WS-Security); URL: http://www-106.ibm.com/developerworks/library/ws-secure.

[18] Microsoft MSDN; Security in a Web Services World: A Proposed Architecture and Roadmap; URL:http://msdn.microsoft.com/ws-security.

[19] Verisign; Enabling Trusted Web Services;URL: http://www.verisign.com/wss.

[20] Microsoft; MS Biztalk Server; URL: http://www.biztalk.org.

[21] Microsoft UDDI Business registry node; URL: http//uddi.microsoft.com.

[22] Microsoft UDDI Business registry node; TestSite; URL: http//test.uddi.microsoft.com.

[23] OASIS; UDDI Version 2.04 API Specification; URL: http://uddi.org/pubs/ProgrammersAPI_v2.htm.

[24] Ing. Pavel Horovčák: IT System; Webové služby – třetí generace internetu; 9/2003, str. 48.

[25] Microsoft ČR; Definice základních součástí .NET;
URL: http://www.microsoft.com/cze/net/basics/whatis.asp.

[26] Jiří Košek; Inteligentní podpora navigace na WWW s využitím XML;
http://www.kosek.cz/diplomka/html/index.html.

# Seznam Příloh