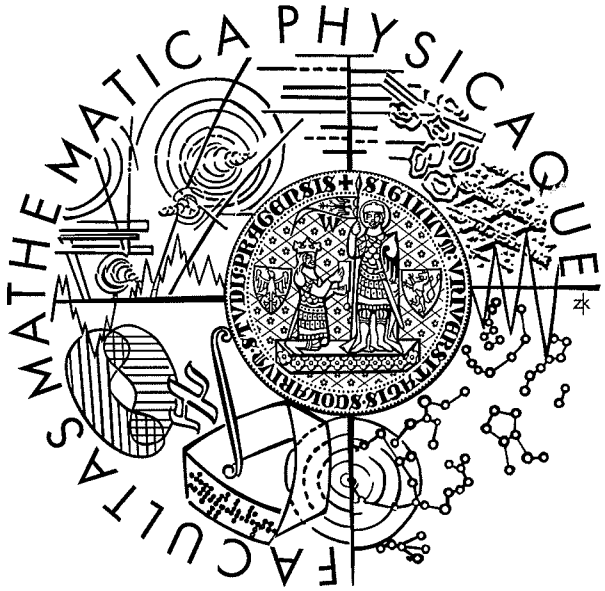
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

Diplomová práce



Boris Lupták

Norma SOAP a její použití v prostředí malých a středně velkých společností

Katedra softwarového inženýrství
Vedoucí diplomové práce: Prof. RNDr. Jaroslav Král, DrSc.

Praha, 2007

Ďekuji vedoucímu diplomové práce prof. RNDr. Jaroslavu Královi, DrSc., za cenné podněty, rady a pomoc při tvorbě této práce.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 15. 4. 2007

Boris Lupták

Obsah

1 Úvod 1
2 Základné pojmy 2
2.1 Formát XML 2
2.2 SOA 3
2.3 Webové služby 6
2.3.1 SOAP 7
2.3.1.1 Základná štruktúra SOAP správy 8
2.3.1.2 Menné priestory v SOAP správach 9
2.3.1.3 SOAP Header element 10
2.3.1.4 SOAP Body Element 11
2.3.1.5 Element SOAP Fault 12
2.3.2 UDDI 14
2.3.3 WSDL 14
3 História a súčasný stav normy SOAP 16
3.1 História SOAP 16
3.2 Súčasný stav SOAP 17
3.3 Bezpečnosť SOAP 19
3.4 Nedostatky SOAP 21
3.5 Navrhované riešenia nedostatkov SOAP 24
3.6 REST 25
3.6.1 Porovnanie SOAP a REST 26
3.7 Vzťah SOAP k middleware 27
3.8 Rozšírenia SOAP z dielne WS-I 28
3.8.1 WS-Basic Profile 29
3.8.2 WS-Basic Security Profile 29
3.8.3 WS-Addressing 29
3.8.4 Ostatné rozšírenia 30
4 SOAP a webové služby v kontexte SOA 31
4.1 Obchodné procesy 31
4.2 Jazyk BPEL 32
4.3 BPEL pre ľudí 33
5 Využitie webových technológií v súčasných projektoch 35
5.1 Webové technológie vo veľkých podnikoch 35

5.2 Webové technológie v malých a stredných podnikoch 38
5.2.1 Príklad použitia SOAP v parkovacom systéme 39
5.2.2 Možnosti integrácie bez využitia webových technológií 41
5.2.3 Príklad integrácie bez využitia normy SOAP 42
5.2.3.1 Predstavenie spoločností 43
5.2.3.2 Základné požiadavky na systém 43
5.2.3.3 Zvolené riešenie 44
5.2.3.4 Analýza zvoleného riešenia 47
5.2.3.5 Alternatívne riešenia 48
5.2.3.6 Zhrnutie 50
5.2.4 Príklad integrácie s využitím normy SOAP 50
5.2.4.1 Počiatočný stav 50
5.2.4.2 Navrhnuté riešenie 52
5.2.4.3 Implementácia 53
5.2.4.4 Záver 54
5.3 Výhody webových služieb 54
5.4 Nevýhody webových služieb 55
5.5 Limitujúce faktory nasadenia webových služieb 56
6 Záver 57

Název práce: Norma SOAP a její použití v prostředí malých a středně velkých společností.

Autor: Boris Lupták

Katedra : Katedra softwarového inženýrství

Vedoucí diplomové práce: Prof. RNDr. Jaroslav Král, DrSc.

e-mail vedoucího: kral@ksi.ms.mff.cuni.cz

Abstrakt: Norma SOAP je základným protokolom webových služieb. Webové služby otvárajú nové možnosti pri interakcii aplikácií s využitím sieťových spojení. V súvislosti s webovými službami je často diskutovaná aj servisne orientovaná architektúra.

Hlavným cieľom tejto práce je ukázať možnosti využitia protokolu SOAP v prostredí malých a stredne veľkých spoločností. Na docielenie tohto zámeru je v práci popísaná norma SOAP a normy jej príbuzné ako aj mechanizmus pomocou ktorého sa protokol SOAP dá rozširovať o nové vlastnosti. Nekoľko štandardných rozšírení je v práci popísaných. Ďalej práca poukazuje na výhody a nevýhody použitia normy SOAP a popisuje príbuzné alternatívne prístupy k prepájaniu aplikácií. Práca ukazuje na konkrétnych príkladoch spôsob použitia technológie webových služieb a normy SOAP.

Súčasťou práce je aj popísanie kľúčových vlastností servisne orientovanej architektúry a ich spojenie s webovými službami. Dôvody, ktoré vedú k potrebe zarovnania webových služieb s obchodnými procesmi a hrubozrnné rozhrania sú tiež v práci diskutované.

Klíčová slova: SOAP, SOA, webové služby

Title: Standard SOAP and its Use in Context of Small and Medium Businesses

Author: Boris Lupták

Department: Department of Software Engineering

Supervisor: Prof. RNDr. Jaroslav Král, DrSc.

Supervisor's e-mail address: kral@ksi.ms.mff.cuni.cz

Abstract: SOAP standard is core protocol used in web services. Web services open new possibilities in application interaction over the network. In connection with web services there has been much discussion of service oriented architecture.

The main goal of this thesis is to show possibilities of SOAP in the context of small and medium businesses. To achieve this goal SOAP and accompanying norms are described as well as the mechanism through which SOAP can be and is extended. Several standard extensions are shown. Furthermore this thesis points out the pros and cons of such usage as well as similar alternative approaches. The usage of SOAP is documented on several real life examples.

Part of this thesis concerns with fundamental principles of service oriented architecture and the connection between web services and service oriented architecture. The need of web services alignment with business processes and coarse grained interfaces are discussed as well.

Keywords: SOAP, SOA, web services

1 Úvod

Súčasný rozmach sieťových riešení otvára nové možnosti nielen pri dnes už klasickej komunikácii človeka s počítačom. Čoraz väčší priestor dostáva takzvaný web pre stroje. Jedná sa hlavne o webové služby[1] s ktorými sa úzko spája pojem Servisne Orientovanej Architektúry (SOA)[2]. Nové technológie poskytujú širšie možnosti v oblasti integrácie systémov. Možnosť integrácie systémov je pri využití webových služieb dosiahnuteľná nielen naprieč programovacími jazykmi a platformami operačných systémov, ale vďaka neustálemu zvyšovaniu výpočtového výkonu zasahuje aj oblasť jednoúčelových zariadení. Tieto nové technológie majú potenciál vytvoriť, pri správnom nasadení, konkurenčnú výhodu.

Preto sa v tejto diplomovej práci zameráme na spojitosť medzi SOA a webovými službami, ako aj na základný kameň webových služieb, a síce komunikačný protokol, ktorý webové služby využívajú – normu SOAP. Na jednej strane zhodnotíme túto normu z technickej stránky, ozrejmíme jej vzťah k starším technológiám middleware a poukážeme na rôzne alternatívne riešenia, ktoré sa snažia zdokonaliť, alebo nahradiť protokol SOAP. Vzhľadom k tomu, že sa SOAP a webové služby úzko spájajú s pojmom SOA, ponúkneme zhodnotenie tejto architektúry. Popíšeme stručnú históriu normy SOAP vrátane motivácie, ktorá viedla k jej vytvoreniu. Následne sa v práci venujeme súčasnému stavu normy SOAP a špecifikáciám, ktoré obohacujú protokol o vlastnosti potrebné hlavne na poli integrácie. Ukážeme jazyky umožňujúce modelovanie obchodných procesov a ich vykonávanie pomocou webových služieb a vysvetlíme prečo je vhodné aby boli IT procesy v súlade s obchodnými procesmi. Poukážeme na aktuálny stav využitia týchto technológií v podmienkach Českej a Slovenskej republiky. Zamyslíme sa nad problémami využitia webových technológií v prostrediach spoločností s rôznymi technologickými zázemiami a identifikujeme bariéry ktoré ovplyvňujú mieru nasadenia webových služieb v širšom kontexte. Počas celej práce je kladený dôraz na aspekty využitia normy SOAP v prostredí malých a stredne veľkých spoločností. Ukážeme na príklade použitie normy SOAP v prostredí firmy s nízkym rozpočtom na IT. Zároveň zhodnotíme alternatívne možnosti prístupu a vyhodnotíme ich efektivitu. Na konkrétnom príklade ukážeme výhody použitia normy SOAP v praxi. Na ďalšom príklade ukážeme riešenie prepojenia dvoch systémov v ktorom neboli použité štandardy SOAP a ukážeme jeho nedostatky.

Na záver práce ponúkneme zhodnotenie normy SOAP ako aj možností jej využitia spolu s dôvodmi pre ktoré jej rozširovanie do nových oblastí nasadenia postupuje len pomaly.

2 Základné pojmy

V tejto kapitole budú stručne vysvetlené základné pojmy ako XML, SOA, SOAP, UDDI, WSDL a ďalšie. Nie je cieľom vysvetliť všetky termíny do hĺbky, ale poskytnúť základný prehľad dôležitých pojmov z pohľadu tejto diplomovej práce.

2.1 Formát XML

Pojem XML[3] je skratkou pre eXtensible Markup Language. Jedná sa o typ formátu dokumentov a je podobný napríklad formátu HTML. Základnou stavebnou jednotkou XML sú takzvané značky (*tags*). Značky nemajú dopredu určený význam ale ten je ponechaný na tvorcovi XML dokumentu. Dokument XML má stromovú štruktúru a má práve jeden koreňový element (*root element*). Stromová štruktúra je významnou vlastnosťou XML dokumentov a v určitých scenároch použitia je limitujúcim faktorom. Napríklad pri reprezentácii dátových štruktúr, ktoré obsahujú ukazovatele a nemajú stromovú štruktúru. Ďalším pravidlom je že každá značka musí mať párovú ukončovaciu značku. Dokument ktorý má správnu syntax sa nazýva dobre formovaný (*well formed*).

Každý dobre formovaný XML dokument sa dá popísať určitou schémou – množinou pravidiel. Táto schéma alebo DTD (*Document Type Definition*) definuje legálne väzby medzi jednotlivými značkami v XML dokumente vrátane ich násobnosti alebo názvov, ako aj umiestnenie jednotlivých značiek v stromovej štruktúre XML dokumentu. Dokument, ktorý je dobre formovaný a naviac je popísaný požadovaným DTD sa nazýva platným (*valid*) XML dokumentom. DTD je súčasťou štandardu XML 1.0. Forma zápisu DTD nie je XML, ale bol vyvinutý špeciálny spôsob zápisu, čo bolo nepraktické najmä kvôli nutnosti vytvoriť ďalší analyzátor. Medzi najväčšie nedostatky DTD patrí napríklad chýbajúca podpora pre menné priestory XML. Ako sa ukázalo štruktúru XML dokumentu je možné výhodnejšie zapísať aj pomocou iného XML dokumentu.

Týmto spôsobom vznikla XML alternatíva k DTD – XML schéma[4]. XML schéma poskytuje širšie možnosti definovania legálnych stavebných kameňov XML dokumentu ako DTD. Pomocou XML schémy, alebo DTD vieme určiť elementy a atribúty ktoré sa môžu vyskytovať v dokumente. Ďalej definujeme jednotlivé elementy, ktoré sú potomkami iných konkrétnych elementov a podobne. Zároveň dokážeme určiť aj typ dát pre jednotlivé elementy. A to buď dáta typu text, číslo, alebo aj presnejšie ako napríklad celé číslo,

nezáporné celé číslo. Môžeme však vymenovať aj množinu prípustných povolených hodnôt a poprípade stanovovať horné alebo dolné limity a podobne.

V dnešnej dobe sa vo veľkej miere na popis XML dokumentu používa XML schéma, ktorá je W3C doporučením od roku 2002. V ďalšom texte budeme predpokladať znalosti XML vrátane menných priestorov.

2.2 SOA

Skratka SOA je akronymom pre Service Oriented Architecture – architektúra zameraná na služby. Vzhľadom na veľkú publicitu SOA sa vyskytuje značné množstvo definícií tejto skratky. Niektoré z definícií sú neúplné, alebo odzrkadľujú chápanie SOA z pohľadu autora tej ktorej definície.

Príklad nejasnej definície môžeme nájsť v[5]. Podľa tejto definície je SOA v zásade kolekcia služieb, ktoré spolu komunikujú. Nejasnosti v definícii SOA sa pokúsilo vyriešiť OASIS, neziskové globálne konzorcium zamerané na vývoj, konvergenciu a adaptáciu štandardov používaných v elektronickom obchodnom styku a sformovalo pracovný výbor pre SOA referenčný model. Výsledkom práce tohto výboru, ktorý „vytvára základný referenčný model aby napomáhal tvorbe systémov založených na servisne orientovanej architektúre", je SOA referenčný model verzia 1.0[6].

Referenčný model je abstraktný rámec, ktorý napomáha k pochopeniu kľúčových vzťahov medzi entitami určitého prostredia. SOA referenčný model definuje SOA nasledovne: „Servisne orientovaná architektúra je prístup k organizovaniu a využívaniu distribuovaných schopností, ktoré môžu byť pod kontrolou rôznych vlastníkov. Poskytuje jednotné prostriedky potrebné k poskytovaniu, vyhľadaniu, interakcii a používaniu schopností umožňujúcich vytvoriť požadovaný výsledok konzistentný s merateľnými predpokladmi.". Medzi kľúčové časti definície patrí hlavne zmienka o rôznych vlastníkoch jednotlivých schopností. Práve táto zmienka už priamo v definícii anticipuje spájanie systémov postavených a prevádzkovaných rôznymi spoločnosťami pravdepodobne na rôznych programových a operačných platformách. Schopnosťou v definícii sa myslí možnosť vykonávať určitú činnosť, ktorá má konzistentný a merateľný výsledok. Predpoklady vyhľadania a interakcie implikujú možnosť formálne popísať jednotlivé schopnosti. Preferovaným popisom je v našom prípade popis, ktorý je strojovo čitateľný.

SOA je z technologického hľadiska podobne ako procedurálne programovanie alebo

objektovo orientované programovanie, koncept, či spôsob vytvárania aplikácii. Ako vidieť z definície OASIS, SOA neoznačuje niektorú konkrétnu techniku alebo programovací jazyk, či protokol, ale spôsob vnútornej štruktúry aplikácie. Pre naplnenie definície musia jednotlivé časti systému spoločne spolupracovať bez ohľadu na ich fyzické rozmiestnenie.

V terminológii SOA sa nehovorí o procedúrach alebo objektoch, ale ako jednotka funkčnosti sa používa pojem služba (*service*). Služby sú navzájom nezávislé ucelenejšie programové celky poskytujúce určitú funkčnosť, ktorá typicky rieši komplexnejšiu úlohu ako procedúra. Služby majú presne definované rozhranie a vykonávajú presne definovanú úlohu. Naviac prevláda intuitívne vnímanie pojmu služieb a tieto sú obvykle previazané s reálnym svetom. To znamená, že vykonanie nejakej služby má dôsledky v reálnom svete, napríklad v podobe odčerpania prostriedkov z bankového účtu a podobne. Práve previazanosť počítačových služieb s ich nepočítačovými verziami má za dôsledok lepšie chápanie IT služieb zo strany riadiacich pracovníkov. Títo automaticky získavajú presnejšiu predstavu o fungovaní IT systému a dokonca ho teoreticky dokážu riadiť. Otázke zarovnania služieb s obchodnými procesmi sa budeme bližšie venovať v kapitole o obchodných procesoch.

Obvyklou požiadavkou na SOA aplikácie je aby mali služby „hrubozrnné" rozhrania (*coarse grained*). Táto požiadavka je podmienená jednak z technologického hľadiska – rozhrania typicky spracovávajú XML dokumenty, ktoré sú prenášané cez sieť. Pri zle navrhnutých rozhraniach, ktoré sú príliš jemné, systém prenáša zbytočne veľké množstvo dát cez komunikačné kanále. Naviac každá správa znamená skonštruovanie XML dokumentu na strane odosielateľa, jeho rozklad a overenie na strane príjemcu. Tieto faktory majú za následok, pri zvýšení počtu správ prenášaných medzi jednotlivými službami, výrazný rast zaťaženia systémov, ktoré tieto správy spracovávajú. Takisto neúmerne narastá zaťaženie prenosových kanálov.

Odhliadnuc od technického hľadiska z analytického pohľadu je výhodnejší systém s menším počtom dobre formovaných rozhraní a v neposlednom rade aj zarovnanie služieb s obchodnými procesmi. Obchodné procesy sú v každej spoločnosti rozdielne, ale boli vytvárané na základe reálnych požiadaviek a väčšinou sú preverené dlhoročnými skúsenosťami. Takéto obchodné procesy majú vo svojej podstate tiež určité rozhrania. Či už sa jedná o predávanie informácií medzi zamestnancami podieľajúcimi sa na danom procese, alebo o interakciu so zákazníkom. V drvivej väčšine štandardných obchodných procesov sú tieto „rozhrania" „hrubozrnné" (*coarse grained*). Táto vlastnosť „hrubozrnnosti" nebola cielene vyhľadávaná, ale bola vyústením optimalizačného procesu. Pre tieto dôvody sa

„hrubozrnné“ rozhrania služieb dostali medzi požiadavky SOA.

Medzi vlastnosti dobre špecifikovaných rozhraní patrí aj dôsledok, že pri zmene implementácie služby nie je, pri dodržaní rovnakej funkčnosti a rozhrania, nutné meniť ostatné služby, ktoré menenú službu využívajú.

Služba môže na dosiahnutie požadovaného výsledku využívať ďalšie služby tak, aby sa optimálnym spôsobom zabezpečilo opätovné použitie už napísaného programového kódu. V architektúre SOA je dôležitým aspektom možnosť voľne viazanej spolupráce jednotlivých služieb. Kooperujúce služby nemusia byť umiestnené na rovnakom počítači, ale na zaručenie ich spojenia môže byť využitý internet či intranet, alebo obecne ľubovoľné iné počítačové spojenie.

Výhodou pri použití SOA je vyššia miera abstraktnosti a tým aj zlepšený náhľad analytikov na riešený problém. Rovnako je prínosom miera zapojenia užívateľov do procesu tvorby aplikácie. Možnosť lepšieho zapojenia užívateľov do špecifikácie aplikácie je daná podobnosťou rozhrania služby s obchodnými procesmi z reálneho života.

Ďalším z dôvodov pre použitie SOA by mala byť aj možnosť promptne reagovať na požiadavky a s využitím už existujúcej kolekcie služieb poskladať výsledné riešenie. Toto môže byť docielené práve možnosťou voľného spájania služieb čím, pri dostatočne veľkej kolekcii už vytvorených služieb, odpadne značné množstvo opätovného programovania.

Dobre navrhnuté služby, ktoré sú v minimálnej miere závislé na centrálnych dátach sú veľmi dobre škálovateľné. Pri preťažení služby je možnosť vytvoriť klon služby na druhom počítači a časť užívateľov presmerovať na túto kópiu služby.

Existujú aj ďalšie možnosti, ktoré prinášajú dobre vytvorené služby. Medzi tieto možnosti spadajú scenáre využitia služieb, pri ktorých sa počíta s odvodzovaním kompozitných webových služieb od vzorových služieb. Odvodená služba má v zásade podobné črty ako jej vzor, ale niektoré jej aspekty môžu byť upravené pre niektorý konkrétny prípad. Takáto služba môže byť jednorázová, alebo môže byť zaradená do depozitára služieb pre opätovné použitie. Takýto scenár je aplikovateľný v prostredí rapídne sa meniacich podmienok. Naviac by potenciálne každé predchádzajúce úspešné riešenie stanoveného problému mohlo byť systémom zaznamenané a týmto spôsobom by sa uchovávalo know-how spoločnosti.

2.3 Webové služby

W3C definuje pojmom webové služby softvérový systém skonštruovaný tak, aby podporoval možnosť spolupráce s jednou, alebo viacerými ďalšími webovými službami bez zásahu človeka za použitia sieťového spojenia. Webové služby sú často len aplikačnými programovými rozhraniami, ku ktorým môže byť pristupované cez sieť, ako napríklad internet, a ktoré môžu byť spustené na vzdialenom systéme. Často sa pod pojmom webová služba rozumie práve služba prístupná cez celosvetovú sieť internet. V istom zmysle internet slúži ako prepojovacia vrstva pre aplikácie.

Takáto definícia zahŕňa množstvo rôznorodých systémov. Existuje viacero pohľadov z technického hľadiska na definíciu webových služieb. V niektorých zdrojoch sa za webové služby označujú aplikačné programové rozhrania, ktoré podporujú vzdialené volanie cez internet. Príkladom by bola celá skupina webových služieb založené na REST prístupe (ako napríklad Yahoo! Web Services). Skupina WS-I považuje za webové služby len služby založené na SOAP a WSLD. V ďalšom texte, ak nebude uvedené inak, sa budú za webové služby považovať tie, ktoré sú založené na SOAP a príbuzných normách.

Kľúčovou vlastnosťou webových služieb je schopnosť prepájať aplikácie vytvorené na rôznych operačných platformách s využitím rôznych programovacích jazykov. Medzi výhody použitia webových služieb patrí aj možnosť opätovného použitia už napísaných programov, takzvaných *legacy* systémov. Toto sa realizuje pomocou vytvorenia rozhrania webovej služby do už existujúcej aplikácie. Druhou možnosťou je, v prípade ak aplikácia podporuje napríklad import a export dát v textovej podobe, vytvorenie adaptéru. Takýto adaptér na jednej strane komunikuje s webovou službou pomocou webového rozhrania a na strane druhej poskytuje rozhranie pre aplikáciu v podobe textového súboru.

Za typické príklady použitia sa považujú scenáre, pri ktorých sa vytvorí kompozitná webová služba. Táto využíva iné už vytvorené webové služby a z ich spojenia vznikne nová funkčnosť. Pre vytváranie takýchto kompozitných služieb je dôležité aby informácie o dostupných službách boli nejakým spôsobom katalogizované. Bez možnosti efektívneho vyhľadania vyhovujúcej už vytvorenej služby sa nemôžu vízie o takejto univerzálnej integrovatelnosti naplniť. Bez katalogizácie dostupných služieb sa potenciálny používateľ o ich existencii nedozvie a nezúžitkuje ich. Namiesto toho, ak je to nutné, vytvorí duplicitnú službu. Význam dôslednej katalogizácie sa ešte viac zvýrazní pri potrebe integrácie rôznorodých systémov v rámci jednej spoločnosti, kde jednotlivé systémy dodávajú rozdielni

dodávatelia. Naviac je vo vnútrofiremnom prostredí dôležité spracovávať aj informácie o používaní jednotlivých služieb ostatnými systémami. Takéto informácie slúžia napríklad k optimálnemu rozvrhnutiu výpočtového výkonu. Na katalogizáciu služieb slúži adresárová služba UDDI, ktorá je popísaná v samostatnej kapitole.

2.3.1 SOAP

Norma SOAP 1.2 „poskytuje definíciu informácie založenej na XML, ktorá môže byť použitá na výmenu štruktúrovanej informácie s kontrolou typov medzi partnermi v decentralizovanom distribuovanom prostredí"[7]. SOAP bol pôvodne používaný ako skratka za *Simple Object Access Protocol*. Vo verzii 1.2 normy SOAP sa už významovo neberie ako skratka, ale len ako označenie protokolu.

Aktuálna verzia SOAP 1.2 poprípade SOAP 1.1 je dnes štandardom pre webové služby. Verziu 1.1 spomíname ako štandard pre webové služby práve kvôli WS – Basic Profile[8], ktorého aktuálna verzia je viazaná na SOAP 1.1. Primárne využitie normy SOAP je pre komunikáciu dvoch aplikácií. Špeciálne sa potom používa pri komunikácii B2B – *business to business* alebo EAI – *enterprise application integration*.

Pre reálne široké uplatnenie musí byť protokol nezávislý na programovacom prostredí a platforme operačného systému. Musí byť flexibilný a založený na štandardných široko rozšírených technológiách. Norma SOAP spĺňa tieto požiadavky a naviac je extenzívne podporovaná zo strany významných softvérových výrobcov a hlavných štandardizačných komisií, akými sú OASIS, W3C, WS-I.

Hlavná úloha normy SOAP je definovanie výmeny informácie prostredníctvom počítačovej siete. Špeciálne sa potom zameriava na zapúzdrenie a zakódovanie informácie do XML dokumentu. Definuje pravidlá na vysielanie a príjem takto zakódovaných správ.

Inštancia SOAP XML dokumentu sa nazýva SOAP správa. SOAP správy sú potom typicky prenášané ako dáta niektorého protokolu, v dnešnej dobe sa jedná hlavne o HTTP alebo HTTPS protokol. Špecifikácia normy SOAP je napísaná tak, aby sa dal použiť takmer ľubovoľný prenosový protokol, napríklad aj SMTP. Verzia normy 1.2 špecificky adresuje väzbu na protokoly HTTP, HTTPS. Naviac norma určuje minimálne požiadavky na prenosový protokol, takže je možné vytvoriť väzby aj s inými protokolmi.

SOAP definuje formálne ako majú byť správy štruktúrované a spracovávané spôsobom, ktorý je nezávislý na použitom programovacom jazyku alebo platforme

operačného systému. Týmto sa zaručuje interoperabilita naprieč celým spektrom programovacích jazykov a operačných systémov.

V ďalšom výklade budeme nazývať SOAP aplikáciami programy, ktoré prijímajú, odosielajú, alebo inak spracovávajú SOAP správy.

2.3.1.1 Základná štruktúra SOAP správy

Správa SOAP, ako bolo povedané, je špeciálnym typom XML dokumentu. SOAP definuje svoju vlastnú XML schému, menné priestory a pravidlá na spracovanie správy. Táto kapitola sa zaoberá štruktúrou SOAP správ a pravidlami na ich spracovanie.

SOAP správy zapúzdrujú prenášanú informáciu podobne ako je tomu v iných protokoloch. Rovnako ako protokol TCP pridáva svoje informácie k prenášaným binárnym dátam, zapúzdruje aj SOAP XML dáta, ktoré sa prenášajú. Podobne ako TCP aj SOAP obsahuje hlavičku, v ktorej sú zapísané potrebné informácie na úspešné prenesenie správy. Významným rozdielom medzi TCP a SOAP je forma samotných prenášaných dát, ktoré sú v prípade TCP ľubovoľné binárne a v prípade SOAP je to XML dokument. Naviac norma SOAP obsahuje mechanizmus, ktorý dovoľuje rozšíriť hlavičku správy o informácie dôležité pre prenos. Existuje W3C note – *SOAP with attachments*[9], ktorá umožňuje prenos okrem XML dokumentu aj ľubovoľných binárnych dát.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:ns1="http://tritonparking.com/cardsIface">
 <SOAP-ENV:Body>
   <ns1:ReadCard>
    <ns1:sn>E0012364558F0</ns1:sn>
   </ns1:ReadCard>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Text 1: SOAP správa

V texte 1 je uvedený príklad SOAP správy, ktorý obsahuje XML element **ReadCard**. V tomto prípade sa predpokladá odpoveď zo strany príjemcu správy v ktorej budú uvedené informácie o karte so sériovým číslom **E0012364558F0**.

Na začiatku správy je uvedená XML deklarácia. Táto určuje typ XML dokumentu a kódovú stránku. Pre SOAP správy nie je povinná a ak je vynechaná predpokladá sa XML verzia 1.0 s kódovaním v UTF-8.

Dôsledná separácia jednotlivých typov elementov je docielená pomocou používania menných priestorov XML dokumentu. V uvedenom príklade je menný priestor SOAP vyznačený tučne a menný priestor samotných dát kurzívou.

Každý XML dokument musí obsahovať koreňový element. V prípade SOAP správy sa jedná o **Envelope**. **Envelope** môže obsahovať nepovinný **Header** element a musí obsahovať povinný **Body** element. Samotné dáta sú obsiahnuté v **Body** elemente. Okrem týchto dvoch elementov nesmie element **Envelope** obsahovať žiadne ďalšie elementy.

Pretože SOAP žiadnym spôsobom nelimituje typ XML dát, ktoré sú zapúzdrené v elemente **Body**, sú SOAP správy veľmi flexibilné. Norma SOAP okrem toho, že povoľuje ľubovoľné XML dáta obsahuje aj špecifikáciu spôsobu, ktorým sa dá tento zapúzdrený XML dokument reprezentovať ako vzdialené volanie procedúry.

Element **Header** môže obsahovať doplňujúce informácie o SOAP správe. Každá z týchto informácii musí byť umiestnená v separátnom elemente, ktorý je priamym potomkom elementu **Header**. Pomocou týchto doplňujúcich informácií sa dá protokol SOAP rozšíriť o rôzne aspekty, ktoré môžu zabezpečovať napríklad transakčnosť a podobné funkcie väčšinou zamerané na kvalitu poskytovanej služby. Tieto rozšírenia nie sú súčasťou štandardu SOAP a v práci budú diskutované v ďalšom texte.

2.3.1.2 Menné priestory v SOAP správach

Menné priestory XML dokumentu sú dôležitou súčasťou SOAP štandardu. SOAP správa môže obsahovať viacero elementov vnorených do **Body** alebo **Header** elementov. Aby sa predišlo kolíziám v názvoch elementov, používajú sa menné priestory, ktoré jednoznačne oddelia tieto elementy od seba. Verzia normy SOAP sa rozlišuje podľa menného priestoru koreňového elementu **Envelope**. Správa patriaca pod normu 1.1 má menný priestor definovaný pomocou URI: "**http://schemas.xmlsoap.org/soap/envelope**" a verzia 1.2 „**http://www.w3.org/2003/05/soap-envelope**". Ak SOAP aplikácia prijme správu, ktorá nepatrí do jedného z podporovaných menných priestorov, tak takúto správu nespracuje. Explicitné prefixovanie elementov patriacich do menného priestoru SOAP je vyžadované.

Každý element v **Header** bloku by mal mať svoj vlastný menný priestor. Tento je obzvlášť dôležitý, lebo sa na jeho základe tieto bloky spracovávajú. Ďalej, aj každý element v **Body** by mal byť prefixovaný patričným menným priestorom.

Vrstvou SOAP budeme nazývať knižnicu, ktorá bola vytvorená aby spracovávala a prenášala SOAP správy. Niektoré SOAP vrstvy extenzívne používajú XML schému na indikáciu dátového typu elementov, iné nie. Tieto rozdiely spôsobujú problém pri spracovávaní správy, ak prijímateľ očakáva, že elementy budú takýmto spôsobom prefixované a odosielateľ ich neprefixuje.

Ako je vidieť použitie menných priestorov je významné nielen pre predchádzanie kolízií názvov elementov, ale hlavne pre použitie správnej verzie a korektného spracovania **Header** a **Body** elementov. Napríklad pri zavedení novej verzie spracovania určitého bloku môžeme túto zmenu reflektovať zmenou menného priestoru v novej verzii bez toho, aby sa ovplyvnilo spracovávanie správ zo starších verzií.

Takéto použitie menných priestorov umožňuje vysokú modularitu SOAP správ. Táto modularita umožňuje aby rôzne časti SOAP správy boli spracované nezávisle na ostatných, čo implikuje aj možnosť vývinu jednotlivých častí nezávisle na ostatných. Napríklad menný priestor elementu **Envelope** sa môže časom meniť s novými verziami špecifikácie, ale samotné aplikačné dáta v elemente **Body** a ich spracovanie môžu zostať zachované.

2.3.1.3 SOAP Header element

Špecifikácia SOAP definuje spôsob, ktorým sú spracované podelementy elementu **Header**. Správa SOAP je odoslaná odosielateľom a na svojej ceste k definitívnemu príjemcovi môže prechádzať cez nula alebo viac prostredníkov. Súčasťou podelmentov elementu **Header** môže byť atribút **Role (Actor)**. Jeho hodnotou je URI jednoznačne identifikujúce funkciu, pre ktorú je element adresovaný.

V značení medzi verziami 1.1 a 1.2 normy SOAP nastala zmena názvu, aj keď význam zostal nezmenený. Vo verzii 1.1 je značenie **Actor** a vo verzii 1.2 sa nazýva **Role**. Táto zmena bola zrejme iniciovaná intuitívnym chápaním slova **Actor** – herec. V skutočnosti môže jedena SOAP aplikácia na ceste SOAP správy od odosielateľa po koncového príjemcu implementovať viacero činností, ktoré boli pôvodne označené ako **Actor**. Zrejme preto sa vo verzii 1.2 zmenil tento názov na **Role** – roľa, čo lepšie vystihuje skutkový stav veci.

Pri spracovaní správy SOAP aplikáciou sa najskôr určia jednotlivé role, v ktorých bude SOAP aplikácia vystupovať. Na toto rozhodnutie môže byť celá správa, vrátane obsahu elementu **Body**, preskúmaná. Norma SOAP 1.2 definuje tri základné role: **next**, **none**, a

ultimateReceiver. Roľa **next** deklaruje, že daný blok je určený pre nasledujúcu SOAP aplikáciu na ceste ku koncovému príjemcovi. Žiadna SOAP aplikácia nesmie figurovať v roli **none**. Jedine koncový príjemca správy figuruje v roli **ultimateReceiver**. Okrem týchto rolí je možné špecifikovať ľubovoľné ďalšie role. Podelementy elementu **Header** sú potom pomocou atribútu **Role** priradené niektorej SOAP aplikácii na ceste medzi odosielateľom a koncovým príjemcom.

Po určení rolí, v ktorých bude SOAP aplikácia vystupovať, sa určia tie bloky z hlavičky SOAP správy, ktoré sú povinné a prislúchajú k niektorej z rolí, ktorú SOAP aplikácia vykonáva. Povinný blok je označený atribútom **mustUnderstand** s hodnotou **1**, alebo **true**. Takéto povinné bloky musí SOAP aplikácia, pre ktorú sú určené, spracovať. Idea je taká, že tieto povinné bloky môžu meniť význam ostatných hlavičkových blokov alebo význam samotných dát uložených v **Body** elemente a preto sa musí zaručiť ich spracovanie. Naviac sa musia spracovať pred ostatnými blokmi SOAP správy. Ak niektorý z povinných blokov nedokáže SOAP aplikácia spracovať, nesmie v spracovaní správy pokračovať a musí vrátiť **SOAP fault** výnimku. Ak aplikácia dokáže spracovať všetky povinné bloky, tak pokračuje ich spracovaním.

Po spracovaní povinných blokov sa spracujú nepovinné bloky. Pri spracovaní sa spracované bloky extrahujú zo SOAP správy. Prostredník musí spracovať jeden alebo viac blokov a môže do **Header** elementu SOAP správy pridávať ľubovoľný počet ďalších podelementov. Žiadna iná SOAP aplikácia, okrem definitívneho príjemcu správy, by nemala manipulovať s obsahom **Body** elementu. Nakoniec, ak sa jedná o prostredníka, sa modifikovaná SOAP správa predá ďalej na ceste ku koncovému príjemcovi.

Vďaka takémuto spôsobu práce je možné pomocou SOAP správ vytvoriť celú škálu modelov posielania správ. Od jednoduchej jednosmernej správy cez synchrónne alebo asynchrónne volania vzdialených procedúr až po sofistikované vzory posielania správ. Špecifikácia SOAP sa nezaoberá žiadnym špeciálnym vzorom a tento je ponechaný plne na implementáciu. To znamená, že súčasťou špecifikácie nie sú žiadne algoritmy, na základe ktorých sa má prostredník rozhodovať, kam správu pošle ďalej.

2.3.1.4 SOAP Body Element

Každá SOAP správa musí obsahovať práve jeden element **Body**. V tomto elemente sa nachádzajú prenášané dáta, alebo správa o chybe – **SOAP fault**. Špecifikácia SOAP

žiadnym spôsobom neurčuje aké dáta môžu byť prenášané, pokiaľ sa jedná o platný fragment XML dokumentu. Dokonca tento element môže byť aj prázdny.

SOAP definuje štyri spôsoby vytvárania správy (RPC/Literal, RPC/Encoded, Document/Literal, Document/Encoded). Vzhľadom na zaručenie interoperability sa odporúča používať spôsob RPC/Literal alebo Document/Literal. Verzie Encoded sú popísané v neslávne známej sekcii 5 verzie SOAP 1.1. Slovo Literal znamená, že sa jedná o XML fragment, ktorý sa dá popísať nejakou XML schémou. Document znamená, že sa jedná o prenos XML dokumentu, alebo presnejšie, že v elemente **Body** sa nachádza XML fragment.

RPC štýl SOAP správy umožňuje aby SOAP správa modelovala vzdialené volanie procedúry, tak ako je toto známe z iných druhov middleware. Pri tomto štýle je obsah elementu **Body** formátovaný ako štruktúra, v ktorej sa nachádza názov procedúry a jej vstupné parametre. Výsledkom takéhoto volania je potom správa, ktorá obsahuje názov procedúry a jej výstupné parametre.

Na rozdiel od Document/Literal štýlu, ktorý nekladie žiadne požiadavky na tvar prijímaných dát, RPC/Literal štýl obsahuje jednoduchú množinu argumentov pre volanú vzdialenú procedúru. Tento štýl SOAP správ uľahčuje, z programátorského pohľadu, tvorbu webových služieb. Na druhú stranu je tento štýl obmedzujúci, lebo nedovoľuje posielať ľubovoľné dáta, ale len také, ktoré sa po sierializácii do XML dokumentu dajú popísať XML schémou. Naviac, ak sa aj dáta dajú popísať schémou, ešte to neznamená, že sa dajú úspešne serializovať. Napríklad kruhové odkazy na dáta a podobné dátové štruktúry je z princípu problematické serializovať a štandardné SOAP knižnice takéto dáta serializovať nedokážu.

2.3.1.5 Element SOAP `Fault`

Norma SOAP venuje veľkú pozornosť generovaniu chybových správ a spôsobu práce s nimi. Jedná sa o mechanizmus pomocou ktorého SOAP aplikácie oznamujú chybu. Chyba sa predáva do predchádzajúceho bodu na ceste medzi odosielateľom a koncovým príjemcom správy. Týmto sa dá zabezpečiť aby o vzniknutej chybe boli upovedomené všetky SOAP aplikácie, ktoré doteraz správu spracovali a umožnilo sa im vykonať prípadné kroky na rollback, alebo podobné operácie. Pri vzniku chyby musí SOAP aplikácia vygenerovať správu so SOAP **Fault**, ale túto správu posiela naspäť len v prípade, ak je použitá dvojcestná komunikácia. V prípade jednosmernej komunikácie sa takto vygenerovaná správa môže uložiť, napríklad do logu, ale nesmie sa posielať naspäť odosielateľovi.

SOAP správa, ktorá obsahuje **Fault** element v **Body** elemente sa nazýva chybová správa. Chybová správa je určitou analógiou s výnimkami v jazyku C++. Takáto správa je vygenerovaná keď nastane chyba. Chyby nastávajú pri nesprávne sformátovanej SOAP správe, odlišných verziách, problémoch pri spracovaní **Header** elementu alebo pri chybách aplikácie.

Pri definovaní **Fault** elementu sú rozdiely medzi verziami SOAP 1.1 a 1.2. Verzia 1.2 zmenila štruktúru **Fault** elementu a pridala do neho doplňujúce informácie.

Chybová správa v **Body** elemente obsahuje práve jeden **Fault** element. **Fault** musí obsahovať **Code** a **Reason** elementy a môže obsahovať **Node**, **Role** a **Detail** elementy.

V **Code** elemente sa nachádzajú podľa kontextu ďalšie podelementy. Vo **Value** podelemente sa môže nachádzať jedna z možností: **VersionMismatch**, **MustUnderstand**, **DataEncodingUnknown**, **Sender**, **Receiver**. V prípade chyby verzie by sa mal do chybovej správy pridať podelement **Upgrade**, ktorý udáva utriedený zoznam URI pre SOAP **Envelope** element podporovaný príjemcom v poradí od najviac preferovanej po najmenej preferovanú verziu.

Pri chybe typu **MustUnderstand** sa pridáva informácia o URI role **Header** bloku, ktorý nemohol byť spracovaný. **DataEncodingUnknown** signalizuje použitie nepodporovaného štýlu kódovania. V prípade verzie 1.2 sa môže jednať len o **encoded** štýl, ktorý je vo verzii 1.2 už len doporučený ale nevyžadovaný.

Chyba typu **Sender** vznikne pri prijatí SOAP správy, ktorá alebo nebola správne formátovaná alebo jej chýbala požadovaná informácia, napríklad v podobe neplatných údajov autorizácie a podobne. Obecne táto chyba indikuje, že správa ako taká bola chybná a odosielateľ sa ju nemá pokúšať posielať opakovane.

Reciever chyba vzniká pri chybe spracovania správy, ktorá nie je priamo spojená s danou SOAP správou. Ak sa odosielateľ neskôr pokúsi odoslať rovnakú správu, tak môže uspieť.

Spolu s týmito preddefinovanými chybovými správami je možné použiť aj iné **Code**. Tieto musia byť súčasťou iného menného priestoru ako SOAP Envelope. Význam a spracovania takýchto chybových kódov sú normou nedefinované. Príkladom rozšírenia chybových kódov môže byť napríklad WS-I Security rozšírenie, ktoré definuje svoje vlastné špecifické chyby.

V ďalších podelemetoch SOAP **`Fault`** sú pridané informácie, ktoré môžu napomôcť k odhaleniu chyby. Jedným z nich je aj textové **`Description`**, do ktorého sa môže pridať textová informácia o vzniknutej chybe.

2.3.2 UDDI

Skratka UDDI [10] znamená Universal Description Discovery and Integration – univerzálny popis vyhľadanie a integrácia. UDDI je adresárovou službou, pomocou ktorej je možné vyhľadať podľa rôznych kritérií v adresári. V adresári UDDI nemusia nutne byť čisto webové služby, ale jeho štruktúry sú navrhnuté natoľko voľne, že môže popisovať napríklad aj adresy kamenných obchodov a popis ich obchodnej činnosti. UDDI je len jednou z možných riešení adresárových služieb.

Pôvodnou myšlienkou v čase prvotného rozmachu webových služieb bola vízia univerzálnej adresárovej služby. Takáto služba by obsahovala informácie o všetkých globálne dostupných webových službách. Webové služby sú z princípu koncipované ako peer to peer spojenia. Adresárová služba by predstavovala centrálny prvok. Takýto centrálny prvok ide vo svojej podstate proti filozofii peer to peer. K problémom univerzálneho UDDI sa pridala aj jeho prílišná všeobecnosť. Tieto a aj ďalšie dôvody prispeli k neúspechu globálneho UDDI.

Vo firemnom prostredí je situácia odlišná. Centrálny prvok architektúry tam nie je takým významným obmedzením a ukazuje sa, že princípy UDDI sa tam s úspechom môžu využiť.

2.3.3 WSDL

WSDL [11] je špecifikácia ktorá definuje spôsob, ktorým sa popisujú webové služby. Skratka WSLD znamená jazyk popisujúci webové služby (*Web Services Description Language*). WSDL popisuje základné informácie o službe – umiestnenie služby, operácie, dátové typy a správy, ktoré webová služba používa. Tieto sa zapisujú vo WSDL dokumente, ktorý má podobu XML dokumentu.

V dokumente WSDL je vyznačený interfejs popisujúci všetky verejne prístupné funkcie, informácie o dátových typoch pre každú zo správ webovej služby, väzba na protokol použitý na prácu s webovou službou a informácia o adrese webovej služby. Dokument WSDL predstavuje popis webovej služby, ktorý je nezávislý na zvolenom programovacom jazyku, alebo platforme operačného systému.

S použitím WSDL dokumentu klient získa všetky potrebné informácie na využitie webovej služby vrátane jej adresy a presných formátov správ, ktoré jednotlivé funkcie webovej služby požadujú a vracajú. S využitím nástrojov na prácu s WSDL môžu informácie v ňom slúžiť ako podklad na automatické generovanie tried na prácu s webovou službou.

3 História a súčasný stav normy SOAP

V nasledujúcej kapitole objasníme dôvody pre vznik normy SOAP a ukážeme jej súčasný stav a spôsob, ktorým sa používa. Priblížime si mechanizmy rozšíriteľnosti protokolu a popíšeme niektoré zo štandardných rozšírení a návrhov na rozšírenie.

3.1 História SOAP

Základ SOAP bol položený v spoločnosti Microsoft v roku 1998. Ako vo svojom článku[12] popisuje pán Don Box, jeden z tvorcov prvotnej špecifikácie SOAP, bolo snahou vytvoriť RPC protokol, ktorý by bol nezávislý na použitej platforme a programovacom jazyku a naviac by nebol filtrovaný firewallmi, respektíve aby voľne prechádzal cez firewally nastavené „štandardným" spôsobom. Vývojári firmy Microsoft vnímali v tom čase ako problém ich doterajších aktivít na poli distribuovaných systémov komplikovanosť konfigurácie sieťovej infraštruktúry. Konkrétne jej nastavenie tak, aby umožňovala prenos správ napríklad pomocou protokolu DCOM v heterogénnom prostredí celosvetovej siete internet. Ďalším kľúčovým problémom bola nekompatibilita a zložitosť jednotlivých typov middleware, ktoré navzájom nespolupracovali. SOAP mal prvoplánovo vyriešiť tieto problémy. Mal ponúknuť integračnú platformu nezávislú na zvolenom dodávateľovi middleware a v neposlednom rade mal za úlohu fungovať bez problémov v prostredí internetu. Posledne zmienená požiadavka bola jedeným z dôvodov prečo bol vybraný protokol HTTP ako jeden z hlavných prenosových prostriedkov. HTTP je protokol povolený na väčšine firemných firewalloch. Takže sa dá povedať že na tomto poli SOAP svoju úlohu splnil. Lenže samotný fakt, že SOAP používa HTTP protokol prispieva k oprávneným obavám o bezpečnosť tejto technológie. Otázka bezpečnosti je natoľko dôležitá, že je jej venovaná samostatná kapitola.

Ďalšou úlohou SOAP bolo poskytnúť základ novo vytváraným webovým službám, ktoré mali reflektovať princípy SOA. Medzi podstatné princípy SOA môžeme zahrnúť práve možnosť vytvárania voľných väzieb medzi jednotlivých službami tak, aby sa vytvorila výsledná služba. Tradične sa na integráciu systémov používali rôzne typy middleware. Jedným z najväčších problémov pri integrácii systémov bola nekompatibilita medzi jednotlivými typmi middleware ako aj medzi rôznymi implementáciami jedného štandardu middleware. Naviac tradičné typy middleware vo svojich špecifikáciách zahŕňajú podporu

zažitých programovacích metód a snažia sa byť príliš univerzálne. Ich tvorcovia sa museli vysporiadať s komplexnými problémami, akými sú distribuovaný garbage collector, čo ich implementáciu činilo obzvlášť náročnou. Naviac napríklad technológia DCOM zakrývala vzdialené volania do takej miery, že pre programátora bolo ich volanie úplne transparentné vrátane predávaných dátových štruktúr. Toto podľa názoru autora diplomovej práce prispelo k vytváraniu príliš komplikovaných systémov, ktoré nielen vďaka transparentnosti s ohľadom na programátora boli navzájom úzko prepojené (*tight coupling*). Naviac, ako už bolo zmienené, nastavenie sieťovej infraštruktúry tak, aby dovoľovala používanie middleware za hranicami lokálnej siete je pomerne komplikovaný proces. Samotní výrobcovia middleware adresovali problém konfigurácie sieťových prvkov a poskytovali rôzne nástroje, ktoré zabezpečovali tunelovanie natívnych protokolov cez HTTP, čím sa umožnila integrácia systémov aj s použitím internetu bez nutnosti rekonfigurácie sieťovej infraštruktúry. Aj keď mnoho technických obtiaží bolo a je riešiteľných, predstavitelia firmy Microsoft sa rozhodli, že sa na prepojovanie systémov vytvorí úplne nový protokol, ktorý uvedené nedostatky odstráni. Pri špecifikovaní normy SOAP sa tvorcovia zamerali na jednoduchosť protokolu a zrejme zámerne zo špecifikácie vynechali množstvo prvkov známych a osvedčených z nástrojov middleware. Autor považuje vynechanie adresácie zo špecifikácie SOAP za kľúčový nedostatok.

V čase svojho vzniku, v roku 1998, stáli pred tvorcami nového protokolu neľahké úlohy a to hlavne v podobe neexistujúceho jazyka pre schémy XML[4]. Preto bol počiatočný vývoj primárne zameraný na definovanie typového systému. Výsledkom vývoja bolo DTD schéma, neskôr prakticky úplne nahradené XML schémou, ktoré bolo vyvinuté v rámci XML skupiny. Dnes sa DTD používa už len zriedkavo.

Po prvotnej špecifikácii SOAP vo verzii 1.1 z roku 2001 pristúpil Microsoft na spoluprácu s IBM na poli webových služieb a následne sa v roku 2002 objavila špecifikácia SOAP vo verzii 1.2. Táto sa od predošlej líši len v detailoch. Samotný fakt spojenia dvoch významných spoločností na poli softvérových riešení prezrádza, že obaja aktéri kladú nemalú váhu na nové technológie.

3.2 Súčasný stav SOAP

V súčasnosti je aktuálna SOAP verzia 1.2.[7] Tradičné softvérové spoločnosti vytvárajú nástroje, takzvané SOAP toolkity, ktoré uľahčujú nasadenie webových služieb. Je vyvinutá podpora na vytváranie a spracovanie SOAP správ vo väčšine programovacích

jazykov. V oblasti serverov sú k dispozícii riešenia tradičných dodávateľov softvérových systémov, ako napríklad IBM Websphere, Sun Microsystems Web Services Developement Pack, ako aj open source alternatívy. Napríklad Apache Tomcat, ktoré ponúkajú kontajnery schopné prevádzkovať webové služby. Medzi hlavné vývojové nástroje serverových komponent webových služieb patrí jazyk Java, a platforma .NET vyvinutá spoločnosťou Microsoft.

Microsoft cez platformu .NET podporuje výrazným spôsobom webové služby. V produktoch akými sú napríklad Microsoft Visual Web Developer je tvorba webových služieb, alebo ich použitie výrazne uľahčené. Konkrétne vo Visual Web Developer Studiu je implementácia webových služieb veľmi jednoduchá. Sú v ňom implementované wizardy na načítanie WSDL popisu služby a v súlade s týmto popisom sa automaticky vygenerujú triedy a funkcie, ktoré transparentne pre programátora využívajú normu SOAP na komunikáciu. Takéto transparentné zakrývanie princípov fungovania má však podľa názoru autora práce negatívne dopady na kvalitu produkovaného softvéru. Vo svojej podstate nie je programátor nútený k znalosti využívanej technológie a v momente kedy sa vyskytnú problémy nie je schopný ich odstrániť, alebo v lepšom prípade ich odstránenie môže trvať neúmerne dlhú dobu.

Veľmi dobrá podpora SOAP je aj v jazyku Java. Napríklad s pomocou knižnice JAX[13] je tvorba a použitie webových služieb výrazne uľahčené. Podpora je veľmi obdobná tej, ktorú ponúka Microsoft vo svojich produktoch. K tejto knižnici sa rovnako ako k produktom Microsoft vzťahuje rovnaká kritika.

Pre ostatné jazyky takisto existuje podpora webových služieb. Firma Borland pre svoje produkty Delphi a Borland Builder už vo verzii 6.0 ponúkala podporu tejto technológie. Zo skúsenosti autora však táto podpora nie je ideálna. Napríklad samotná knižnica VCL[14], tvoriaca jadro systému knižníc od Borlandu, obsahuje chyby. Jednou z nich je nesprávne serializovanie boolean premenných. Namiesto true a false sú serializované ako True a False. Môže sa zdať že sa jedná o formalizmus, ale zlou serializáciou nespĺňa knižnica štandard a znižuje sa interoperabilita s ostatnými produktami iných výrobcov. Borland ďalej využíva XML parser obsiahnutý v operačnom systéme. Naviac sa za každých okolností snaží používať najvyššiu dostupnú verziu. Najnovšia verzia XML parsera obsiahnutá vo windows XP service packu 2[15] má odlišné chovanie, ako predošlé verzie. Toto odlišné správanie má za následok nefunkčnosť automaticky generovaných webových služieb, ktoré nie sú schopné poskytnúť ich WSDL. Spoločnosť Borland naviac ani neuvoľnila opravy týchto chýb. Našťastie existujú

neoficiálne opravy[16]. S novšími verziami nemá autor skúsenosti, ale je predpoklad, že takéto a podobné problémy sú v nich už vyriešené.

Odhliadnuc od riešení na podporu webových služieb mierených na enterprise klientelu, existuje celá škála open source knižníc. Podľa autora je pre jazyky C a C++ asi najvhodnejšou gSOAP [17]. Táto knižnica je aktívne vyvíjaná. Jej prístup k podpore SOAP je cez kompilátor. Tento kompilátor na základe WSDL dokumentu vygeneruje hlavičkové súbory obsahujúce dátové štruktúry prislúchajúce k WSDL dokumentu. Kompilátor WSDL umožňuje aj opačný proces, kedy na základe definície triedy vygeneruje WSDL dokument. Na základe hlavičkových súborov vygenerovaných z WSDL je potom druhý kompilátor schopný vygenerovať C++, alebo C kód, ktorý dovoľuje programátorovi jednoduchým spôsobom knižnicu používať. Druhý zmienený program je schopný generovať kód pre server, alebo klient webovej služby. Štandardy rozšírení normy SOAP, ako sú WS-Security, alebo WS-Addressing sú v knižnici podporované cez mechanizmus plug-in modulov. Zmienené dva moduly sú k dispozícii. Knižnica sa autorovi práce javí ako dobre riešená a jednoduchá na používanie. Pravdou je, že sa v jednoduchosti použitia nevyrovná nástrojom integrovaným priamo do vývojového prostredia, a vyžaduje určité hlbšie znalosti o fungovaní SOAP. Na druhej strane takéto znalosti sú, podľa autora, esenciálne pre tvorbu systémov využívajúcich SOAP. Nevýhoda v podobe počiatočnej nutnosti oboznámiť sa s technológiou je napokon vyvážená získanými znalosťami.

V oblasti ladenia webových služieb existuje celá paleta špecializovaného softvéru. Pre jednoduchšie aplikácie si programátor úplne vystačí s logmi obsiahnutými priamo vo webovom serveri. SOAP správy sú dobre čitateľné aj pre človeka, čo kladným spôsobom ovplyvňuje ladenie komunikácie založenej na SOAP.

3.3 Bezpečnosť SOAP

Norma SOAP a pridružené štandardy predstavujú pomerne nové riešenie. Ako také samozrejme nie je preverené časom. Naviac je rozumné domnievať sa, že tak ako s každou novou technológiou sa aj v prípade SOAP objavia aj nové techniky, ktoré budú využívať jej zraniteľnosti.

Je potrebné si uvedomiť, že pomocou webových služieb a SOAP sa často otvárajú staré aplikácie svetu. Sprístupnením starých aplikácií sprístupňujeme aj ich chyby. Zároveň sa objavili aj nové útoky [18], ktoré pomocou SOAP a špeciálne formovaných správ útočia na zraniteľnosti napríklad XML parsera. V niektorých prípadoch môže nastať pretečenie

zásobníka a útočník tak môže spustiť svoj škodlivý kód. Takéto útoky by mali byť zamedzené s využitím technológie Data Execution Prevention. Táto technológia vyžaduje podporu priamo v procesore a zároveň aj v operačnom systéme. V budúcnosti by mohla výrazným spôsobom obmedziť množstvo úspešných útokov na základe pretečenia buffera a následného spustenia škodlivého kódu z dátovej oblasti pamäti.

Medzi ďalšie problémy na ktoré sa pozabúda patrí aj fakt, že pokiaľ za pomoci webových služieb sprístupňujeme aplikáciu, tak musíme mať na pamäti, že de facto sprístupňujeme aj všetky zdroje, ktoré má daná aplikácia k dispozícii. V prípade ak sa vyskytuje v aplikácii chyba, tak sa táto potenciálne môže dať využiť ku kompromitácii dát, ktoré má aplikácia k dispozícii. Často môžu byť takýmito aplikáciami sprístupnené databázy v ktorých sa môžu nachádzať citlivé údaje. Jednoduchý príklad známej techniky SQL injekcie ponúka [19].

Najpoužívanejším protokolom pre SOAP správy je HTTP, alebo HTTPS. Tento protokol je používaný cez štandardný port 80. Ako vo svojom článku[20] píše pán Schneier, tento fakt sťažuje korektné nastavenie fierwallov. Pán Prescod má rovnaký názor a vo svojom článku[21] poukazuje na to, že štandardné nastavenie firewallov umožňuje SOAP správam priechodnosť. Týmto sa obchádzajú firemná politika na využívanie prostriedkov siete a osoby za ňu zodpovedné. Ak by sa použil nejaký proprietárny port, tak by zodpovedná osoba musela tento port explicitne povoliť, čo je v kontraste s prípadom, kedy sa na zakázanie nežiadúcej komunikácie musí meniť štandardné nastavenie. Medzi najvýraznejších propagátorov SOAP patrí firma Microsoft. Používanie SOAP cez port 80 je v rozpore s víziou Billa Gatesa, že musia viesť IT priemysel na novú úroveň dôveryhodnosti[22]. V článku [22] je zmienené, že ak by Microsoft chcel dostáť týmto vyhláseniam, mal by stiahnuť implementáciu SOAP, ktorá tuneluje cez HTTP za účelom obchádzania firewallov.

Ďalším aspektom bezpečnosti SOAP je fakt, že bezpečnosť nie je súčasťou SOAP štandardu. Bezpečnosť, alebo spôsoby použitia kľúčových bezpečnostných protokolov je súčasťou štandardu WS-Security[23]. Je pravda, že na základné zabezpečenie môžeme použiť protokol HTTPS. Toto riešenie sa však nie vždy a všade hodí, a naviac nerieši niektoré scenáre použitia. V podstate poskytuje len zabezpečenie dát z bodu do bodu a znemožňuje použitie niektorých vlastností SOAP, ako napríklad možnosť predávania zakódovaných správ cez viacero prostredníkov, alebo možnosť autentifikácie za použitia špecializovaného prostredníka. Tieto možnosti použitia sú pritom esenciálne pre univerzálne nasadenie webových služieb.

Na riešenie nedostatkov samotného štandardu SOAP sa dnes špecializuje organizácia Web Services Interoperability (WS-I) . Jej snahou je dodať chýbajúce štandardy pre webové služby. Táto iniciatíva mala za následok vytvorenie celej skupiny návrhov na štandardy. Obecne sa tieto návrhy na štandardy, alebo doporučenia označujú ako WS-*. WS-I nie je jedinou organizáciou zaoberajúcou sa rozšíreniami normy SOAP a príbuzných noriem. Samotné návrhy týchto noriem vychádzajú hlavne od spoločností zaoberajúcimi sa problematikou webových služieb. Vzhľadom na množstvo nepresne špecifikovaných alebo úplne chýbajúcich vlastností normy SOAP bolo nutné vytvoriť adekvátne množstvo doplňujúcich štandardov. Bezpečnosťou SOAP obsahu SOAP správ sa zaoberá štandard od OASIS – WS-Security [23].

3.4 Nedostatky SOAP

Aj keď sa samotná špecifikácia SOAP už niekoľko rokov nemení, menia sa výklady jej správneho použitia (*best practices*). Tieto sú zhrnuté v takzvanom Basic Profile[8] a vo veľkej miere sa presadzujú hlavne cez toolkity, ktoré uľahčujú nasadenie webových služieb. Toolkity sú aplikácie schopné z definície triedy, package, alebo nejakého ekvivalentu v niektorom z podporovaných jazykov priamo vygenerovať WSDL dokument popisujúci príslušnú triedu. Takto vygenerované WSDL obsahuje aj rozhrania na vzdialené volanie metód a spôsob serializácie parametrov k príslušným funkciám. Toolkity podporujú aj opačný proces, kedy sa z WSDL generuje reprezentácia SOAP správy v konkrétnom jazyku vrátane prototypov funkcií na jej spracovanie. V praxi sú programátori cez extenzívny marketing presvedčovaní, že je správne už prvotné WSDL vygenerovať za pomoci toolkitu. Táto praktika však v princípe ide proti logike tvorby systému. V podstate naznačuje, že programátor má vytvoriť najskôr triedu a až na jej základe si nechá vygenerovať popis jej rozhrania. Naviac pri takomto postupe sa musí nutne v popise služby odraziť použitý programovací jazyk a jeho natívne dátové štruktúry. V skutočnosti by mal byť proces presne opačný – a síce navrhnúť rozhranie, napísať príslušné WSDL a k nemu následne implementovať triedu. Samozrejme tento logicky správny postup je možný, ale nie príliš sa hodí do marketingovej kampane, lebo vyžaduje hlbšie znalosti z oblasti štandardov akými sú WSDL.

Pochopenie týchto štandardov predstavuje počiatočnú bariéru pre využívanie webových služieb a z marketingového hľadiska nie je vhodné dopredu upozorňovať na náklady spojené s adopciou technológií. Marketing oveľa viac presadzuje stanovisko v ktorom vykresľuje jednoduchosť použitia bez potreby nových znalostí. Toolkity v niektorých

menej triviálnych prípadoch generujú WSDL dokumenty, ktoré nie sú kompatibilné so systémami od iných výrobcov. V určitých prípadoch je obtiažna spolupráca systémov na bázi jazyka Java a systémov bežiacich na platforme .NET. V prípadoch kedy WSDL vyprodukované niektorým toolkitom zlyhá sú programátori tak povediac ponechaní napospas. Odrazu sú nútení k aktivitám ktoré nikdy predtým nerobili a nerozumejú im. Sú nútení ručne editovať generované WSDL tak, aby prinútili tieto dve platformy k vzájomnej spolupráci. Príkladom dokazujúcim tieto tvrdenia je preferovaný spôsob SOAP kódovania, ktorý sa od RPC encoded zmenil na Document Literal. Druhý zmienený spôsob hovorí v podstate to, že telo SOAP správy je tvorené fragmentom XML dokumentu. Čo je významne odlišné od prvotného doporučenia, kedy boli preferované interakcie na báze Remote Procedure Call so SOAP kódovaním. Motiváciou k zmene doporučenia bolo hlavne zlepšenie interoperability knižníc podporujúcich SOAP.

Väčšina webových serverov nekontroluje platnosť prijatej SOAP správy na základe XML schémy prislúchajúcej k tej ktorej konkrétnej SOAP správe. Táto praktika sa zaužívala kvôli vysokej časovej a pamäťovej náročnosti tejto kontroly a v záujme zvýšenia priepustnosti systému. Aj keď sa nejedná v podstate o nedostatok SOAP ale o nedôslednosť pri jeho využívaní, autor práce si myslí, že je dôležité na tento fakt upozorniť.

SOAP je XML dokument s platnou XML schémou. Jazyk schém však poskytuje aj možnosť vytvorenia takých dátových typov, ktoré nie sú natívne pre samotný programovací jazyk. Rovnaké tvrdenie platí aj opačným smerom. XML schéma nepodporuje vytváranie odkazov na dáta. Výsledkom použitia odkazov na dáta môže byť napríklad vytvorenie kruhového zoznamu dát a podobných nestromových štruktúr. Napriek tomu niektoré knižnice na podporu SOAP sú schopné serializovať aj takéto zložitejšie dátové za pomoci trikov. Takto serializované správy sú potom len ťažko použiteľné v SOAP toolkitoch iných dodávateľov, čo znižuje možnosť spolupráce.

Špecifikácia SOAP explicitne hovorí, že SOAP je nezávislý na zvolenom komunikačnom protokole. Avšak jediným dnes reálne použiteľným protokolom je HTTP, poprípade HTTPS.

V špecifikácii SOAP nie je žiadnym spôsobom vyriešené adresovanie. Toto je ponechané na mechanizmy protokolu nad ktorým SOAP beží. Už samotný fakt, že SOAP nie je vytvorený pre nejaký konkrétny protokol a neexistencia modelu adresácie napovedá, že v budúcnosti pri adaptácii, alebo prechode na iný protokol môžu nastať problémy. Ďalej, napríklad v prípade zlyhania webovej služby SOAP server vygeneruje SOAP fault správu.

Táto je doručená v prípade HTTP k správnemu príjemcovi cez mechanizmus odpovede servera. Avšak nie je žiadnym spôsobom definované ako by sa mal implementovať mechanizmus odpovede obecnejšie. Tento je vždy úzko spätý s použitým protokolom. Pritom v niektorých prípadoch by bolo vhodné, aby sa SOAP fault správa odoslala na iné miesto ako odosielateľovi, poprípade aby sa odoslala na viacero miest. Napríklad by sa chybová správa mohla poslať e-mailom správcovi systému.

SOAP, ako už bolo spomenuté, naschvál neobsahuje veľké množstvo vlastností. Úplne chýba podpora akejkoľvek bezpečnosti. Pre úplnosť dodáme, že jediným spôsobom zabezpečenia je použitie protokolu HTTPS. Takéto zabezpečenie je však výhradne zabezpečením z bodu do bodu a prakticky nedovoľuje použiť SOAP v rozmere v ktorom tvorcovia zamýšľali – využívať prídavné informácie v hlavičke SOAP správy bez odkrytia obsahu správy.

Ku kritike SOAP toolkitov môžeme pridať jeden konkrétny príklad. Jedná sa o prenos dátového typu xsd::dateTime. Tento primitívny dátový typ je špecifikovaný ako reťazec predstavujúci dátum a čas a pripúšťa špecifikovanie časovej zóny, pre ktorú je daný čas lokálnym časom. Do problému sa pridáva nielen časová zóna ale aj letný a zimný čas. Autor práce testoval predávanie dátumu a času do a z webovej služby, ktorá bola implementovaná s pomocou Visual Basic s knižnicou .NET 2.0. Klientom bola aplikácia napísaná v jazyku C++ s knižnicou gSOAP kompilovaná v Borland C++ Builder. Na strane klienta je knižnica gSOAP, ktorá pre dátový typ dateTime požaduje UNIX time v UTC. Takýto čas prevedie pomocou štandardných funkcií localtime a príbuzných na lokálny čas počítača a tento serializuje spolu s informáciou o časovej zóne počítača do SOAP správy. Problémom, ako sa ukázalo, bolo už samotné prevedenie lokálneho času do UTC času. Knižnice, ktoré tento prevod zabezpečujú, nesprávne rozoznávajú letný a zimný čas. Zrejme sa jedná o verzie knižníc staršieho dáta dodaných spoločnosťou Borland ku kompilátoru Borland C++ Builder, ktoré sa nesprávajú korektne. Na druhej strane je treba sa zamyslieť nad samotnou podstatou letného a zimného času, ktorý je určený v rámci zákona. Takže knižnica by musela mať minimálne možnosť nastavenia správnych dátumov pre posun. Na strane servera sa jednalo o produkt spoločnosti Microsoft, ktorý správne rozoznal formát dátumu a času a tento čas predal do funkcie, ktorá implementovala webovú službu. V prípadoch, kedy časová zóna prijatého dátumu a času nezodpovedala časovej zóne servera, bol čas správne posunutý, tak aby zodpovedal lokálnemu času. Avšak v momente, kedy bol zo strany klienta poslaný čas v UTC a tento fakt bol signalizovaný v SOAP elemente, tak už server tento čas z neznámeho dôvodu nenormalizoval do lokálneho času, ale ponechal ho v UTC. Naviac v .NET 2.0

pribudol atribút dátového typu dátum, ktorý určuje, či je reprezentovaná hodnota dátumu a času v UTC alebo v lokálnom čase. Deserializačná funkcia, ktorá prevádza čas zo SOAP správy však tento argument ignoruje a nenastavuje ho. Z uvedeného vyplýva faktická nemožnosť použitia vstavaných funkcií na prácu s dátovým typom dateTime. Na strane Visual Basic totižto neexistuje možnosť ako zistiť, či je čas zadaný v UTC, alebo v lokálnom čase. Na strane gSOAP je zrejme z dôvodu použitia Borland C++ Builder nekonzistentný dátum a čas v UTC forme, práve kvôli zle implementovaným časovým posunom pre letný a zimný čas. Jediná kombinácia, ktorá by mohla používať vstavané funkcie by vyžadovala prenos v dátumu a času v UTC so správne vyznačenou časovou zónou a s dohodou, že funkcia na strane Microsoft bude očakávať čas v UTC. Takáto kombinácia je závislá práve na dohode o význame parametra a neexistuje spôsob ako ju efektívne vynucovať. Podľa názoru autora je jediná rozumná možnosť ako pracovať s parametrami typu dateTime napísanie vlastných serializačných a deserializačných funkcií, lebo tie, ktoré sú k dispozícii v oboch toolkitoch sú defakto nepoužiteľné.

3.5 Navrhované riešenia nedostatkov SOAP

Komplexné riešenia horeuvedených nedostatkov by vyžadovali zmenu samotného protokolu SOAP. V snahe o rýchle riešenie najväčších nedostatkov sa zrodila organizácia WS-I (*Web Services Interoperability*). Táto organizácia zvolila namiesto úpravy samotného protokolu jeho doplnenie jednak cez nadstavbové špecifikácie a jednak cez vydanie doporučení, ktorými sa popisujú správne prípady použitia. V ďalšom texte je organizácii WS-I a doporučeniam ktoré vytvorila venovaná samostatná kapitola.

Jedným z riešení je SOAP vôbec nepoužívať a na miesto neho používať REST prístup. Prístupu REST je venovaná samostatná kapitola.

Určitá časť nedostatkov SOAP pramení zo spôsobu, ktorým sa SOAP používa, menovite zo SOAP toolkitov a vrstiev. Páni Loughran a Smith z Hewlet Packard navrhli pozmeniť stávajúci SOAP vrstvu v jazyku Java a vytvoriť novú vrstvu[24]. Interne ju pomenovali Alpine SOAP vrstva. Alpine by mal byť formovaný okolo XML (*XML centric*) a bez bližších znalostí princípov SOAP by nemal byť použiteľný. Návrh samotnej aplikácie by mal postupovať logickým spôsobom od návrhu rozhrania k jeho implementácii a nie naopak, ako to z praxe pozorujeme. Ďalej Alpine podporuje len mapovanie základných dátových typov a odvodenín na SOAP správy, čím na jednu stranu komplikuje fázu návrhu rozhrania, ale na stranu druhú zaisťuje oveľa vyššiu mieru interoperability. Zároveň Alpine automaticky

validuje prichádzajúce SOAP správy príslušnými XSD. V podstate sa jedná o zjednodušenie funkcií knižnice JAX.

3.6 REST

Pojem REST použil vo svojej dizertačnej práci pán Fielding. Jedná sa o Representational State Transfer, ktorý je softvérovou architektúrou pre distribuované systémy hypermédií, ako napríklad World Wide Web [25]. Hypermédium je logickým rozšírením pojmu hypertext, v ktorom sa prelína grafika, text, audio a video aby vytvorili obecne nelineárne informačné médium [26].

Princípy na ktorých je založený REST sa dajú zhrnúť do nasledujúcich bodov:

- Stav a funkčnosť aplikácie je rozdelená do zdrojov. (*resources*)
- Každý zdroj je jednoznačne adresovateľný za použitia univerzálnej syntaxe, ktorá je určená na použitie v odkazoch hypermédií.
- Všetky zdroje zdieľajú jednotné rozhranie na prenos stavu medzi klientom a zdrojom
- Obsahuje obmedzenú množinu dobre definovaných operácií
- Obmedzená množina obsahových typov (*content type*), nepovinne podporujúca „code-on-demand"
- Protokol má vlastnosti
  - Klient / Server
  - Bez stavový
  - Cacheovateľný
  - Vrstvený

Ako sa zdá REST je značne reštriktívny. Zrejme najväčšou reštrikciou je jeho bezstavovosť v zmysle, kedy sa vyžaduje aby každá otázka z klienta na server obsahovala všetky informácie potrebné k porozumeniu požiadavky. Pri spracovaní požiadavky sa nesmie použiť žiaden kontext uložený na serveri. Toto má za následok mimo iné aj zvýšené nároky na prenosovú infraštruktúru, lebo každá správa musí obsahovať celkový stav aplikácie, ktorý je uložený na klientovi. Tento spôsob zvyšuje nároky na zachovanie konzistencie dát v aplikácii. Nároky na prenos sa ale dajú kompenzovať pomocou cache. Na druhej strane

prístup REST využíva dobre overené technológie webu. Pomocou REST sa dajú vytvárať plnohodnotné webové služby a aplikácie, ktoré môžu spĺňať kritéria SOA.

V prospech prístupu REST oproti norme SOAP hovoria nižšie vstupné náklady – nie je to úplne nová technológia ale návod ako efektívnejšie využívať stávajúce programové prostriedky a infraštruktúru. (REST rozhranie v on-line kníhkupectve amazon – 80% dotazov, SOAP rozhranie zvyšných 20%)

V praxi REST webová služba pracuje s XML rovnako ako SOAP. Ale REST nedefinuje žiaden formát samotnej správy. To čo je pri protokole SOAP umiestnené v samotných dátach, alebo v blokoch elementu **`Header`**, sa dáva v REST prístupe do hlavičky protokolu HTTP (GET, PUT, POST, DELETE). Jediným kvázi štandardným zabezpečením je použiť HTTPS. Ak je nejaké ďalšie zabezpečenie potrebné, potom sa implementuje proprietárne.

V každom prípade je REST tým, že vo veľkej miere využíva štandardné technológie oveľa priateľskejší voči programátorom, čím si získal na popularite. Zároveň do veľkej miery využíva súčasné nástroje na správu sieťovej prevádzky, čím sa znižujú náklady na zavedenie do praxe.

Fielding vo svojej dizertačnej práci ukazuje ako pri návrhu REST boli adresované požiadavky na flexibilitu riešenia a dokazuje, že tieto požiadavky sa s úspechom dajú splniť pri správnom použití prístupu REST.

3.6.1 Porovnanie SOAP a REST

Oba protokoly sú v podstate rovnaké v tom čo umožňujú, zjednodušene povedané podporujú výmenu XML dokumentov medzi dvoma aplikáciami pomocou počítačovej siete. Tam ale podobnosť končí. Definícia SOAP umožňuje prácu nad ľubovolným protokolom, aj keď dnes preferovaný je HTTP, REST je špecifický použitím výhradne HTTP (alebo HTTPS).

Norma SOAP je založená na štandardoch W3C, je prepracovanejšia, často až do miery ktorá môže byť kontraproduktívna. Na druhej strane REST je založený na koncepte využitia stávajúcich štandardov a nie je formálne špecifikovaný.

Prístup REST je minimalistický, na jeho správu a odladenie stačia štandardné nástroje ako sú webový prehliadač. SOAP je komplexnejší so štandardizovanými možnosťami rozšírenia. Správy SOAP vyžadujú väčší procesorový čas na spracovanie. REST je menej

náročný na prostriedky.

Interoperabilita je v prípade SOAP do značnej miery ovplyvnená štandardami a doporučeniami, ktoré sa neustále vyvíjajú ako odozva na problémy pri integrácii. Tieto nové štandardy sa presadzujú do praxe proprietárnymi produktami akými sú rôzne pomôcky a aplikačné rámce (*frameworks*) na zjednodušenie práce so SOAP a pridruženými normami. Tieto sú do značnej miery orientované na niektorý konkrétny jazyk, alebo platformu a v konečnom dôsledku znižujú schopnosť spolupráce so systémami ostatných výrobcov. Pre prístup REST k webovým službám sa používajú len nástroje, ktoré sú štandardné a široko rozšírené. XML je dnes štandardom na každej počítačovej platforme a má vytvorené knižnice na jeho spracovanie v ľubovoľnom jazyku. A vzhľadom k tomu, že XML v prípade REST prístupu obsahuje len samotné dáta a nie sú na neho kladené žiadne dodatočné obmedzenie v podobe hlavičiek a podobne, nie sú ani na jeho konštrukciu potrebné žiadne dodatočné nástroje.

Ako sa ukazuje SOAP prevláda hlavne vo firemnom sektore, kde sa úspešne uplatňuje na poli integrácie rôznych systémov. REST sa presadzuje mimo poľa veľkých firemných aplikácií, kde sa s úspechom využíva jeho jednoduchosť a povedomie programátorov o všetkých štandardoch, ktoré využíva.

3.7 Vzťah SOAP k middleware

Middleware sa definuje ako „softvérová vrstva ležiaca medzi aplikáciami a operačnými systémami na oboch stranách sieťového prepojenia distribuovaného počítačového systému"[27]. Norma SOAP spĺňa túto definíciu a môžeme ju preto považovať za typ middleware.

Middleware sa delí na kategórie: Vzdialené volanie funkcií (*Remote procedure call – RPC*), Middleware orientovaný na prenos správ (*Message oritented midlleware – MOM*), *Object Request Broker – ORB*, Prístup k dátam s využitím SQL (*SQL-oriented Data Access*).

Medzi najznámejšie middleware staršieho dáta patria napríklad CORBA, alebo DCOM. CORBA[28] je vyvíjaná neziskovou organizáciou Open Management Group (OMG). Pán Henning vo svojom článku[29] popisuje dôvody pre ktoré CORBA neuspela. Medzi hlavné dôvody zaraďuje jednak technické aspekty štandardu CORBA, akými sú užívateľsky nepriateľské API, ktoré viedli k vysokej chybovosti v programoch, alebo celková veľkosť štandardu, ktorého časti neboli nikdy implementované ani na potvrdenie konceptu. Pán

Henning pripisuje tieto technické nedostatky hlavne spôsobu ktorým bola CORBA vytváraná. OMG je konzorcium, ktorého členmi sa mohla stať ľubovoľná spoločnosť. Členovia OMG patria do dvoch skupín – dodávatelia riešení a zákazníci. Na prijatie niektorého návrhu sa hlasovalo a na základe tohto hlasovania bol návrh zaradený do štandardu. Týmto spôsobom sa prijali návrhy, ktoré boli nekoncepčné a mali technické vady. Produkty podporujúce štandard CORBA neimplementovali všetku funkcionalitu ktorú zahŕňal štandard a v prípade nejasností alebo technických problémov prišli s vlastným riešením, ktoré nebolo súčasťou štandardu. Výsledkom boli problémy pri komunikácii programov postavených na odlišných implementáciách štandardu. Pán Henning v závere článku vyjadruje myšlienku, že webové služby, čo sa týka spôsobu prijímania štandardov opakujú rovnaké chyby, ktoré viedli k postupnému úpadku štandardu CORBA.

Firma Microsoft nikdy nepodporila iniciatívu CORBA, ale prišla s vlastným riešením middleware, DCOM (*Distributed Component Object Model*)[30]. Technológia DCOM nebola úspešná hlavne vďaka proprietárnosti riešenia. DCOM pracoval výhradne na platforme Windows. Dnes sú k dispozícii aj open source implementácie. Samotný Microsoft považuje technológiu DCOM za zastaranú a jej miesto zaujali webové služby v platforme .NET.

Keď porovnáme SOAP so zmienenými technológiami vidíme, že SOAP oproti DCOM zabezpečuje lepšiu interoperabilitu. V porovnaní s CORBA sú webové služby jednoduchšie použiteľné. Webové služby, ako už bolo spomenuté používajú protokol HTTP, čím sa uľahčuje ich nasadenie v prostredí internetu. Na druhej strane, webové služby použitím jazyka XML výrazne zvyšujú požiadavky na výpočtový výkon ako aj prenosovú infraštruktúru. Zatiaľ webové služby nepodporujú zďaleka všetky funkcie známe z CORBA alebo z DCOM.

3.8 Rozšírenia SOAP z dielne WS-I

Organizácia WS-I (*Web Services Interoperability*) nemá za cieľ meniť podobu špecifikácie protokolu SOAP, ale cez mechanizmus rozšírení tento protokol doplniť o potrebnú a žiadanú funkcionalitu. Organizácia WS-I sa delí na šesť pracovných skupín, kde každá z nich je zodpovedná za riešenie určitých špecifickejších problémov. Výsledky práce týchto skupín majú prispieť ku všeobecnej interoperabilite systémov. Doterajšia prax je taká, že pri integrácii, kde bolo nutné použiť niektorú z vlastností dnes špecifikovanými WS-I sa na príslušnom protokole dohodli tvorcovia aplikácií, ktoré spolu komunikovali. Tento postup síce riešil akútny problém s prepojením aplikácií, ale vznikalo množstvo navzájom nekompatibilných protokolov riešiacich rovnaký problém. Pri každej ďalšej integrácii bolo

nutné prispôsobovať sa novým partnerom, čo je v priamom rozpore s proklamovanými výhodami protokolu SOAP.

3.8.1 WS-Basic Profile

Ako už samotný názov napovedá jedná sa o doporučenie od organizácie WS-I. Momentálne je aktuálna verzia Basic Profile (BP) 1.1[8] z 10. apríla 2006. Jedná sa o sadu doporučení, vyjasnení a interpretácií tých špecifikácií, ktoré podporujú interoperabilitu. Konkrétne BP obsahuje doporučenia pre samotné správy (normu SOAP), popisovanie webových služieb (norma WSDL), ako aj na objavovanie služieb (UDDI).

V oblasti posielania správ je aktuálna verzia BP mierená na SOAP verziu 1.1. Explicitne zakazuje použitie encoded správ, ktoré boli zdrojom najväčších problémov pri zabezpečovaní interoperability.

SOAP aplikácia môže explicitne prehlasovať zhodu s BP. Takéto tvrdenie je v samotnej správe vyznačené príslušným URI cez mechanizmus prehlásenia zhody[31]. Prehlásenie o zhode je v kompetencii tvorcu príslušnej SOAP aplikácie. WS-I dodáva sadu testov, ktoré majú pomôcť pri prehlásení zhody. Ak príslušná SOAP aplikácia prejde úspešne týmito testami, tak je možné ju prehlásiť za zhodnú s BP.

3.8.2 WS-Basic Security Profile

Podobne ako Basic Profile sa jedná o sadu doporučení a spresnení štandardov týkajúcich sa bezpečnosti, konkrétne WS-Security špecifikácie. Zameranie je na mechanizmy transportnej vrstvy, SOAP elementy, hlavičky SOAP správ atď. Bližšie podrobnosti sa nachádzajú v [32]. Finálna verzia tejto špecifikácie je úplne nová, konkrétne z 30. apríla 2007.

3.8.3 WS-Addressing

WS-Addressing[33] je zahrnuté vo working drafte WS-Basic Profile 1.2. Jedná sa o mechanizmus adresácie SOAP správ, ktorý je nezávislý na zvolenom protokole. Špecificky sa dá potom určiť protokol pre samotnú komunikáciu a separátne napríklad pre SOAP Fault správy. Tým pádom sa dá špecifikovať, že chybové SOAP správy budú prenášané napríklad pomocou SMTP protokolu a na zvolenú adresu a podobne.

Ako bolo viackrát uvedené je SOAP protokol úzko naviazaný na HTTP protokol, aj

keď samotná špecifikácia tvrdí, že je nezávislá na prenosovom protokole. V skutočnosti však transportný protokol do veľkej miery ovplyvňuje samotné SOAP správy. Práve preto, že SOAP špecifikácia sa nezaoberá spôsobom doručenia správy, ale spolieha sa na mechanizmy prenosového protokolu. Tým pádom je celá SOAP špecifikácia do veľkej miery závislá na vytvorených väzbách medzi transportným protokolom a SOAP správou.

WS-Addressing je W3C návrhom ako pridať podporu adresácie do štandardu. Zvolený spôsob je cez rozširujúce mechanizmy `Header` elementu SOAP správy. Návrh má dopady aj na vlastnosti samotného SOAP protokolu. Napríklad pán Davis v článku[34] rozoberá dopady tohto návrhu na samotnú podstatu protokolu SOAP. Konkrétne sa jedná o synchrónnu a asynchrónnu komunikáciu. Pán Davis dôvodí, že práve možnosťou explicitnej voľby prenosového protokolu všetky SOAP aplikácie podporujúce tento návrh automaticky musia podporovať asynchrónne spôsoby výmeny správ. Dokonca, už nie je možné s istotou o nejakej službe tvrdiť, že využíva len synchrónne volania. Pretože hlavičky vo WS-Addressing môžu zmeniť vzor otázka – odpoveď na asynchrónnu komunikáciu pomocou explicitnej špecifikácie konkrétneho servera, kam sa má odpoveď doručiť.

WS-Addressing je návrhom, ktorý prináša zaujímavé možnosti pre využitie protokolu SOAP. Autor sa domnieva, že taká základná vlastnosť protokolu akou je model adresovania mala byť zakotvená priamo v štandarde SOAP. WS-Addressing zapĺňa určitú medzeru v SOAP štandarde, ktorej neadresovanie si v minulosti vyslúžilo nemalú kritiku.

3.8.4 Ostatné rozšírenia

Okrem zmienených rozšírení existuje celá paleta ďalších. Tieto sú väčšinou iniciatívou niektorej zo známych spoločností, akými sú BEA, Microsoft, IBM a ďalšie. Tieto spoločnosti medzi sebou návrhy rozšírení väčšinou nekoordinujú a výsledkom sú duplicitné „štandardy" riešiace tie isté, alebo veľmi podobné problémy. Základný prehľad o WS štandardoch ponúka [35]. Nie je možné vzhľadom na množstvo týchto štandardov ich v tejto práci popísať.

4 SOAP a webové služby v kontexte SOA

Webové služby založené na SOAPe majú relatívne blízko k SOA, ale musíme si uvedomiť, že SOAP neimplikuje SOA a naopak. SOAP a webové služby môžu byť len jedným z nástrojov ako dosiahnuť SOA. V tejto kapitole popíšeme možnosti využitia technológie SOAP na tvorbu systémov servisne orientovanej architektúry, ako aj dôvody prečo majú byť rozhrania webových služieb zarovnané s obchodnými procesmi.

Za pomoci normy SOAP je možné vytvoriť systém, ktorý nemá prvky architektúry SOA a naopak systém s architektúrou SOA nemusí používať protokol SOAP. Pre architektúru SOA nie je záväzný žiaden z protokolov a praktiky architektúry SOA sa dajú porušiť vytvorením rozhraní za pomoci SOAP, ktoré napríklad porušujú pravidlo granularity.

4.1 Obchodné procesy

Obchodný proces je množina spojených aktivít, ktoré vytvárajú hodnotu transformovaním vstupov na výstupy s vyššou hodnotou[36]. Vstupy aj výstupy môžu byť artefaktami, alebo informáciami. Transformácia môže byť vykonávaná ľuďmi, strojmi, alebo oboma. Poznáme tri základné typy obchodných procesov. Manažérske procesy ovládajú činnosť spoločnosti. Typické príklady zahŕňajú napríklad strategický manažment. Operačné procesy sa primárne podieľajú na vytváraní hodnoty v spoločnosti. Príkladmi operačných procesov sú napríklad nákup, výroba, marketing, alebo predaj. Posledným typom obchodných procesov sú podporné procesy. Tieto sa priamo nepodieľajú na tvorbe hodnoty, ale vhodným spôsobom ju podporujú. Príkladmi by mohli byť účtovníctvo, IT podpora a iné. Aktivity sú časťami obchodného procesu, ktoré nevyžadujú žiadne rozhodovanie.

Obchodnými procesmi sa dajú popísať takmer všetky činnosti vykonávané v spoločnosti. Operačné obchodné procesy sa priamo zúčastňujú na tvorbe zisku spoločnosti. Podporné procesy zabezpečujú funkcie, ktoré síce priamo nevytvárajú zisk, ale sú pre prevádzku spoločnosti nepostrádateľné. Manažérske procesy zabezpečujú z dlhodobého hľadiska konkurencie schopnosť a prežitie spoločnosti. Je zrejmé, že optimalizáciou ktoréhokoľvek procesu sa znížia náklady alebo sa zvýši zisk, či sa zabezpečí zlepšenie strategického plánovania, čo prispeje v dlhodobom meradle k udržaniu, alebo zvýšeniu rentability spoločnosti. Preto by optimalizácia mala patriť k strategickým cieľom podniku. Medzi techniky optimalizácie sa môže zaradiť aj automatizácia. Pomocou automatizácie sa dá

docieliť zvýšenie konkurencieschopnosti spoločnosti.

Penetrácia IT do všetkých oblastí podnikateľských činností sa neustále zvyšuje. Spoločnosti tvoria akýsi organizmus, ktorý nemôže prosperovať bez svojho okolia. Automatizácia interakcií nielen v rámci jednej spoločnosti, ale aj naprieč celým spektrom dodávateľov a odberateľov umožní ďalšie zvýšenie efektivity práce a konkurencieschopnosti. Realizovanie obchodných procesov za pomoci programového vybavenia priamo prispieva k automatizovaniu niektorých aktivít, alebo dokonca celých procesov. Keďže obchodné procesy priamo popisujú náplň činnosti spoločnosti, čím bližšie programové vybavenie kopíruje tieto procesy, tým lepšie a presnejšie sa tieto manuálne vykonávané procesy dajú programami nahradiť. Touto myšlienkou sa zaoberá architektúra SOA, kde služby v ideálnom prípade korešpondujú s jednotlivými aktivitami z obchodných procesov.

Z uvedeného zarovnania obchodných procesov s webovými službami vyplývajú určité výhody. Okrem už spomenutej automatizácie sa vďaka zarovnaniu webových služieb s obchodnými procesmi priblížia pojmy manažérov spoločnosti s pojmami, ktoré v súvislosti s projektom používajú programátori. Títo vďaka dobre navrhnutým rozhraniam získajú dobrý prehľad o fungovaní systému, ktorý majú vyvinúť. Z tohto pochopenia dokážu profitovať pri vytváraní a testovaní systému. Manažéri sa vďaka princípu SOA dokážu lepšie orientovať v programovom vybavení spoločnosti. Jednotlivé webové služby kopírujú pre manažérov známy model obchodných procesov. V prostrediach, alebo v situáciách kde je treba často meniť obchodné procesy a reagovať tak na novovzniknuté požiadavky môže byť aktívna účasť manažéra na formovaní obchodného procesu nevyhnutnosťou.

Vytvorenie webových služieb, ktoré korešpondujú s aktivitami v obchodom procese je prvým krokom k modelovaniu obchodného procesu. Ďalším krokom je správne pospájať jednotlivé služby tak, aby vznikol požadovaný kompozitný proces. Jedným zo spôsobov ako spojenie docieliť by bolo priamo naprogramovať kompozitnú službu, ktorá by predstavovala celý proces. Existuje však aj riešenie, kde za pomoci konfigurácie vzniká spájanie webových služieb. Jedným z možných riešení je jazyk BPEL[37].

4.2 Jazyk BPEL

Pomocou jazyka BPEL (*Business Process Execution Language*) sa formálne popisujú obchodné procesy. Tento popis je naviac strojovo spustiteľný. Implementácií jazyka BPEL existuje celá škála. Väčšinou sa jedná o implementácie veľkých spoločností akými sú Oracle, BEA a podobne, ale nájdu sa aj open source projekty.

Jazyk BPEL vznikol v spolupráci IBM a Microsoft zlúčením ich dvoch jazykov – WSFL (*Web Service Flow Language*) a XLANG. Obsahuje základné programovacie konštrukty, akými sú premenné, podmienky a cykly. BPEL dokáže manipulovať XML dokumenty, a vyvolávať webové služby. Samotný program v jazyku BPEL môže byť vystavený ako webová služba.

V reálnom nasadení sa popíšu procesy v spoločnosti, tieto sa rozdelia na dielčie aktivity. Pri rozdeľovaní na aktivity sa prihliada k univerzálnosti a znovu použiteľnosti jednotlivých aktivít. Tieto aktivity sa implementujú a za pomoci webových služieb sa prezentujú. Pomocou nástroja na orchestráciu webových služieb sa tieto pospájajú tak aby vytvorili ucelený obchodný proces.

Ak je súčasťou obchodného procesu rozhodovanie na základe nejakých kritérií, toto sa jednoducho popíše za pomoci podmienok v jazyku BPEL. Premenné vstupujúce do podmienok sa extrahujú z XML pomocou XPath a XQuery.

Z popisu jazyka BPEL je vidieť, že sa veľmi dobre hodí na popísanie a spustenie procesov, ktoré pozostávajú z jednotlivých webových služieb a pri ktorých sú dané jasné pravidlá pre prevedenie. V realite ale často potrebuje do obchodného procesu za určitých podmienok vstúpiť manažér a proces ovplyvniť svojimi rozhodnutiami. Tento typ interakcie medzi užívateľom a systémom nie je súčasťou špecifikácie BPEL, čo je jeho veľkou nevýhodou pri reálnom nasadení. Je ale možnosť ako upraviť štandard BPEL tak, aby podporoval aj komunikáciu s užívateľmi. Návrh na úpravu jazyka sa nazýva BPEL4people.

Na oblasť webových služieb je mierený štandard BPEL4WS , ktorého verzia 1.1 [37] z mája 2003 je druhým verejným návrhom BPEL4WS špecifikácie.

4.3 BPEL pre ľudí

BPEL pre ľudí (*BPEL4People*) [38] je návrh ako do jazyka BPEL zahrnúť interakcie s človekom.

V reálnych obchodných procesoch hrá významnú úlohu ľudský faktor. Ak by sme si ako príklad zobrali objednanie tovaru, tak za každú objednávku musí byť určená zodpovedná osoba. Majme napríklad dve firmy, dodávateľa a odberateľa. Nech má dodávateľ implementované prijímanie objednávok cez webovú službu. A nech má odberateľ implementovaný komplexný systém skladového hospodárstva, ktorý je schopný automaticky chýbajúci tovar objednať u dodávateľa. Aj keď sa zdá, že proces objednania tovaru môže byť

plne automatizovaný, v skutočnosti musí za každú objednávku ručiť poverená osoba na strane objednávateľa. Ak by systém kvôli chybe, alebo kvôli zlému nastaveniu objednal miesto jedného auta tisíc áut, musí byť za takúto chybnú objednávku niekto zodpovedný.

Zodpovednosť nie je jediným prípadom v reálnych obchodných procesoch, kde je potrebný dozor človeka nad systémom. Návrh BPEL4People označuje a popisuje sedem rôznych štandardných scenárov, ktoré zachytávajú úlohu človeka v obchodných procesoch.

Základným scenárom sú aktivity ľudí. Tieto sa dajú predstaviť tak, že človek v podstate vykoná činnosť namiesto webovej služby. Rozšírením tohto scenára je scenár pri ktorom človek iniciuje spustenie obchodného procesu. Počas behu obchodného procesu je potrebné tento proces monitorovať a v prípade, keď je to nutné, musí mať manažér procesu možnosť do procesu zasiahnuť. Napríklad ak sa proces zastaví, lebo systém nebol automaticky schopný prideliť určitú činnosť konkrétnej osobe. V tomto prípade je nutné, aby manažér procesu do neho vstúpil a ručne priradil vykonanie činnosti niektorej dostupnej osobe. Iný príklad zahŕňa vypršanie času určeného na vykonanie činnosti. Ak je činnosť súčasťou zložitého obchodného procesu, ktorý bol spustený počas dlhej doby a jeho plnením sa spotrebovalo nemalé množstvo času a prostriedkov, nebolo by najvhodnejšie takýto proces zastaviť, lebo zastavením by sa stratila všetka doterajšia práca vykonaná na danom procese. Výhodnejšie je, ak administrátor takéhoto procesu môže zvoliť najlepšie riešenie, napríklad predĺžením doby na vykonanie činnosti.

Návrh obsahuje aj rozšírené scenáre použitia, akými sú princíp štyroch očí. Pri tomto scenári systém vyberie z množiny oprávnených ľudí náhodne niekoľkých a tí nezávisle vykonajú tú istú úlohu. Napríklad rozhodnutie o pridelení pôžičky. Ich odpovede sa porovnajú a proces pokračuje ak nastala zhoda. Princíp eskalácie je ďalším popisovaným scenárom. Pri ňom je definovaná hierarchia zodpovednosti a ak sa niektorá z úloh neplní, ako bolo naplánované, tak eskaluje – je pridelená nadriadenej osobe z hierarchie. Nominácie popisujú scenár pri ktorom sa na základe pravidiel prideľuje vykonanie úlohy konkrétnej osobe. Posledný zmieňovaný scenár je zreťazené vykonávanie činností. Tento scenár sa uplatní v prípade, keď nejaká činnosť pozostáva z rady krokov, ktoré sú vykonávané rovnakou osobou. Namiesto klasickej interakcie, kde si človek po splnení kroku úlohy obnoví zoznam jemu pridelených úloh, je vhodné aby sa pridelenie ďalšieho kroku vykonalo automaticky.

Popísané scenáre použitia tvoria základnú množinu reálnych interakcií človeka v obchodnom procese a spolu s automatizovanými webovými službami by mali postačovať na definovanie reálnych obchodných procesov.

5 Využitie webových technológií v súčasných projektoch

V tejto kapitole je stručne zmapované využívanie webových služieb v súčasných projektoch, ktoré je rozdelené podľa veľkosti podniku. Ďalej sú rozobrané dôvody súčasného stavu v oblasti využitia webových služieb v malých a stredných podnikoch. Na záver kapitoly sú uvedené príklady použitia pre segment malých a stredných firiem a analýza konkrétneho nasadenia integračného riešenia v praxi.

Na problematiku využitia webových služieb je potrebné pozerať sa z dvoch uhlov pohľadu. Z perspektívy zadávateľov projektov – zákazníkov, ako aj z pohľadu dodávateľov riešení. Je zjavné, že zadávatelia projektov – zákazníci si pri nákupe softvérového riešenia stanovujú podmienky, ktoré musí riešenie splniť. Rozsahovo veľké projekty zadávajú zvyčajne veľké spoločnosti. Ich manažéri sledujú strategické ciele a sú zvyčajne dobre oboznámení s technologickými možnosťami, ktoré sú k dispozícii. Toto tvrdenie je obzvlášť pravdivé v spoločnostiach s dobrým technologickým zázemím a s vyššími rozpočtami v oblasti IT. Strategické ciele spoločnosti sú ciele dlhodobé a dobre vybudovaný informačný systém prináša konkurenčnú výhodu. Pri takýchto projektoch je predpoklad, že budú preferované riešenia, ktoré ponúkajú možnosti integrácie s ostatnými, už vybudovanými alebo plánovanými systémami.

Správanie menších zákazníkov je spravidla diametrálne odlišné. Len zriedkavo si menšie spoločnosti nechávajú vytvoriť informačný systém na mieru. Oveľa častejšie sa uspokoja s takzvanými krabicovými riešeniami. Autor sa v praxi často stretáva pri takomto type zákazníkov so základnými neznalosťami v oblasti IT. Neexistencia vízie a neznalosti špecifík IT odvetvia často vedú k nesprávnym požiadavkám na systém a tým pádom k nákupu nevhodných softvérových riešení.

5.1 Webové technológie vo veľkých podnikoch

V dnešnej dobe je využívanie webových technológií na prepájanie rôznorodých systémov na vzostupe. Jedná sa väčšinou o nové projekty pri ktorých sa s úspechom využívajú časti alebo aj celé pôvodné aplikácie. Pri mapovaní využitia webových technológií autor práce použil kontakty na pracovníkov spoločností IBM, SAP, Cleverlance a Sun Microsystems. Zrejme vzhľadom na pracovnú vyťaženosť oslovených pracovníkov bolo získané len dve stanoviská k problematike použitia webových technológií.

Autor použil pri oslovení kompetentných zamestnancov hore zmienených spoločností prístup pri ktorom vypracoval otázky o využití SOAP v projektoch s ktorými sa stretli, alebo na ktorých pracovali. Tieto otázky boli mierené jednak na využívanie SOAP ako aj na znalosti pracovníkov z oblasti webových služieb, konkrétne štandardov týkajúcich sa SOAP a ich využitia.

Pracovník spoločnosti SAP, pán Jakub Souček, vo svojej odpovedi na otázky uviedol jeho skúsenosti s webovými technológiami v spoločnosti SAP. Pán Souček pracuje na pozícii konzultanta integračnej platformy SAP NetWeaver Exchange Infrastructure (XI). Vyjadril svoj súkromný názor, že spoločnosť SAP vzhľadom na rozsah jej aplikačnej funkcionality v oblasti riadenia a dlhú dobu jej vývoja, cca od sedemdesiatych rokov, nie je spoločnosťou zameranou na technologické novinky. Spoločnosť SAP preferuje dlhodobou praxou overený aplikačný model. Naviac v projektoch s ktorými pán Souček prišiel do styku sa zvyčajne integrujú aplikácie staršieho dáta, ktoré obsahujú import a export dát len v textovej podobe. Zákazníci nemajú záujem investovať prostriedky do úpravy týchto pôvodných aplikácií. Na druhú stranu je spoločnosť SAP pripravená aj na poli webových služieb a v rámci XI ponúka aj prepojenia pomocou webových služieb. Samotný produkt XI používa SOAP interne na komunikáciu a celý systém XI sa tak chová ako webová služba.

Z názoru pána Součka je zrejmé, že webové služby sa pri veľkých integračných projektoch v Českej republike využívajú. Je tiež vidieť, že integrácia pôvodných aplikácií do systému SAP sa vytvára za pomoci adaptérov, ktoré z textových dát vygenerujú SOAP správu a tá je následne spracovaná XI engine. Návratové hodnoty sa spätne transformujú zo SOAP správ do textovej podoby vhodnej pre tú ktorú pôvodnú aplikáciu.

Medzi ďalšie body, ktoré autor práce s pánom Součkom diskutoval patrili aj nové normy z dielne WS-I. S týmito bol pán Souček oboznámený, ale v samotnom SAP systéme sa nevyužívajú. Nie je vylúčené, a autor práce sa domnieva, že tieto štandardy budú do produktov spoločnosti SAP implementované, keď budú overené praxou a zákazníci si ich vyžiadajú.

Ďalšou skupinou spoločností, kde sa s úspechom webové služby presadili sú hlavne sektory bankovníctva, štátnej správy, výrobných podnikov, alebo obecnejšie sa jedná o spoločnosti ktoré majú vysoké IT rozpočty. Príkladom je prípadová štúdia od British American Tobaco[39]. Z uvedenej prípadovej štúdie nepriamo vyplýva napríklad zaujímavý fakt, že SAP zaostával vo využití nových technológií v porovnaní s konkurenciou. Tento fakt je, ako uviedol pán Souček zo spoločnosti SAP, do značnej miery spôsobený internou

politikou SAPu. Konkrétne, v rámci vnútrofiremnej politiky sa nepreferujú nové technológie, ale riešenia sa navrhujú s využitím technológii overených časom. Takisto drvivá väčšina integrácii SAPu je so systémami staršieho dáta a zvyčajne prebieha formou importu, alebo exportu textových súborov. Autor práce sa domnieva, že príčiny tohto konzervatívneho prístupu môžeme hľadať aj vo výsostnom postavení spoločnosti SAP na poli obchodných riešení. Vďaka postaveniu na trhu nie je spoločnosť SAP nútená hľadať a využívať nové technológie na integráciu na úkor kvality. V podstate systém SAP obsahuje takmer všetky moduly, ktoré si zákazník môže priať. Avšak zákazníci, ako je vidieť v uvedenej prípadovej štúdii, sú schopní vyvinúť taký tlak na SAP, že tento rozšíril moduly webových služieb. Naviac by pre firmu SAP bolo zrejme finančne nevýhodné a zložité podporovať všetky možné nové technológie, ktoré sa na trhu objavia. Postavenie na trhu umožňuje spoločnosti SAP vyčkať, pokým sa situácia vykryštalizuje a následne úspešné technológie ponúknuť aj svojim zákazníkom aj za cenu určitého oneskorenia.

Iným faktorom pre ktorý nemusí byť výhodné pre firmy veľkosti SAP, aby podporovali architektúru SOA s webovými službami je aj zvýšenie konkurencie. Ak sa niektoré moduly SAP dajú relatívne jednoducho nahradiť modulmi s rovnakou, alebo lepšou funkčnosťou od iných výrobcov zvyšuje to tlak na SAP a jeho cenotvorbu.

Druhým osloveným bol pán Matej Polák. Pán Polák pracuje v spoločnosti Cleverlance na pozícii J2EE developer. S webovými službami sa stretol na viacerých projektoch veľkých spoločností, na ktorých pracoval v rámci outsourcingu. Z jeho skúseností z projektov pre telekomunikačné spoločnosti vyplýva, že ich požiadavky na použitie integračných technológií sú rôzne a výraznou mierou závisia na dobe začatia projektu integrácie. Požadované technológie sú potom poplatné dobe vzniku projektu. Napríklad pri jednom projekte boli použité webové služby s centrálnym proprietárnym prvkom. Tento prvok v podobe centrálnej brány *(gateway)* slúži na prepájanie jednotlivých systémov spôsobom, kedy jeden systém vygeneruje SOAP správu a predá ju tejto bráne. Brána v sebe obsahuje logiku na základe ktorej rozhodne o príjemcovi správy a túto mu prepošle. Brána nie je žiadnym štandardným riešením, ale bola vyvinutá špeciálne pre potreby zákazníka. V dnešnej dobe by zrejme bolo použité niektoré zo štandardných riešení orchestrácie webových služieb.

Pri inom projekte sa pán Polák stretol s preferovaním Java Message System. Ak bolo treba integrovať podsystém, ktorý nebol schopný pracovať s JMS, tak sa pristúpilo k riešeniu cez tabuľky databázy. V tomto riešení sa výzva v podobe XML zapísala do jedného stĺpca tabuľky, kde si ju prijímací systém vyzdvihol. Na jej základe potom vygeneroval výstupnú

správu, ktorú vložil do príslušného druhého stĺpca, odkiaľ si ju prečítal jej príjemca. Riešenie prepojenia systémov v tomto projekte bolo staršieho dáta a použité technológie túto skutočnosť odrážajú.

Z uvedených poznatkov oboch pánov vyplýva, že sa webové služby využívajú hlavne v projektoch novšieho dáta. Pri projektoch starších sa javí ako neekonomické meniť ich fungovanie.

5.2 Webové technológie v malých a stredných podnikoch

Ako bolo ukázané, použitie webových služieb vo veľkých spoločnostiach je dnes realitou. Veľké spoločnosti však negenerujú väčšinu pridanej hodnoty v národnom hospodárstve. Ako vyplýva z [40] je v malých a stredných podnikoch (MSP) (do 250 zamestnancov) zamestnaných 70,8 % zamestnaného obyvateľstva na území Slovenskej republiky. Podobné čísla vykazuje aj správa o vývoji malého a stredného podnikania (MSP) a jeho podpore v roku 2005 [41] na území Českej republiky. Zároveň sektor MSP vykazuje podľa [41] 53,43 % podiel účtovnej pridanej hodnoty na celku v ČR. Celkový počet MSP je 99,98 % z celkového počtu podnikov. Podniky MSP v roku 2005 investovali celkovo 52,88 % všetkých investícií v ČR. Z uvedených faktov vyplýva, že malé a stredné podniky tvoria významný trh. Väčšia časť vytvorenej pridanej hodnoty pochádza z malých a stredných spoločností. Otázkou zostávajú dôvody pre ktoré sa webové služby nepresadzujú aj v projektoch týchto firiem. Pritom niektoré z projektov sú, alebo majú potenciál, čo do počtu integrovaných systémov, konkurovať známym projektom z oblasti veľkých firiem.

Webové služby z technologického hľadiska nepredstavujú výraznú výzvu. Naviac ponúkajú štandardizované riešenie, ktoré má potenciál zaručiť bezproblémovejšiu integráciu s ďalšími systémami. Vďaka možnosti ľahšej integrácie nie je zákazník odkázaný na jedného konkrétneho dodávateľa systému. Požadovaný systém je, vďaka možnosti integrácie, možné poskladať z menších systémov od rôznych dodávateľov. Žiaden monolitický systém, ktorý sa snaží pokryť veľkú škálu IT požiadaviek firmy z princípu nemôže poskytovať najlepšie výsledky vo všetkých aspektoch. Napríklad je nesprávna domnienka, že softvér zahraničnej firmy bude mať spracované české, alebo slovenské účtovníctvo do takých detailov ako lokálni dodávatelia. Napriek tomu sa autor v praxi stretáva s postojom zákazníkov, ktorí apriori preferujú riešenia zahraničných spoločností, bez ohľadu na špecifiká podmienok daného trhu.

5.2.1 *Príklad použitia SOAP v parkovacom systéme*

Príklad ignorovania špecifickej legislatívy môžeme nájsť napríklad v implementácii parkovacieho systému od firmy Skidata[42]. Táto nespĺňa slovenské zákony v tom, že pokladňa tohto systému nedokáže vytvárať daňové doklady, ktoré slovenské zákony vyžadujú. Tento legislatívny nedostatok systému sa dá riešiť napríklad ručným vypisovaním náhradných daňových dokladov pracovníkmi obsluhy. Ale v prípade automatických pokladní ani táto možnosť nepripadá do úvahy. Tento nedostatok, často bagatelizovaný zo strany dodávateľa, by mal takýto systém úplne diskvalifikovať v slovenskom prostredí. Zaujímavým faktom je, že aj napriek tomu, že systém nevyhovuje legislatíve je v prevádzke minimálne v jednom konkrétnom prípade v Bratislave.

Rovnaké legislatívne prekážky sú kladené na ľubovoľný pokladničný systém napríklad aj v Poľsku. Česká republika doposiaľ nezaviedla povinnosť registračných pokladní. Pre české prostredie je tento problém zatiaľ nepodstatný. Otázkou je či sa v budúcnosti aj Česká republika nepriklorí k praktikám vyžadujúcim registračné pokladnice. V prípade napríklad Poľska sa, vzhľadom na veľkosť trhu dá očakávať, že spoločnosť Skidata investuje prostriedky a zosúladí účtovný modul s požiadavkami miestnej legislatívy.

V slovenskom prostredí sa tento konkrétny problém spravidla rieši integrovaním špeciálneho hardvérového zariadenia, ktoré je certifikované štátnou skúšobňou. Toto hardvérové zariadenie vykonáva funkcie registračnej pokladnice. Pripája sa spravidla medzi tlačiareň dokladov a počítač a ovláda sa špeciálnymi sekvenciami znakov. Na jeho výstupe je správne vytvorený daňový doklad a naviac si zariadenie udržuje informácie potrebné pre tlač daňových uzávierok a elektronickú tlačovú pásku, z ktorej je za pomoci servisného programu možné vytlačiť kópiu ľubovolného daňového dokladu, ktorý bol z pokladne vytlačený. Správne zaokrúhľovanie sumy na daňových dokladoch, alebo daňových uzávierkach, vytváranie kópie dokladov, či už na presne definované elektronické médium, alebo papier sú príklady z mnohých podmienok pridelenia daňového kódu pokladnice, tak ako ich stanovuje vyhláška 55/1994 z. z. o spôsobe vedenia evidencie tržieb elektronickou registračnou pokladnicou. Obdobné pravidlá platia v rôznych štátoch sveta a nie sú žiadnym spôsobom štandardizované.

Takéto, na prvý pohľad jednoduché, hardvérové riešenie je vhodné pre obmedzený počet pokladní. Naviac v každom ďalšom legislatívnom prostredí, kde je potreba nasadiť pokladničný systém je nutné často významným spôsobom upravovať nielen rozhranie, ale v

niektorých prípadoch aj samotnú logiku aplikácie. Uvedený problém sa dá, za určitých predpokladov, riešiť veľmi efektívne napríklad pomocou webových služieb.

Na tomto mieste je nutné uviesť, že dnešné jednoúčelové zariadenia, ako napríklad automatická pokladňa, sú často postavené na platforme osobných počítačov s operačným systémom Linux, Windows CE, alebo podobnom. Disponujú sieťovým pripojením, plne grafickou obrazovkou, atď. To znamená, že sa jedná o plnohodnotné počítače, ktoré sa programujú spravidla v jazyku C++. Java v tomto prípade nie je kvôli svojim vysokým nárokom na pamäť zatiaľ preferovaným riešením. Napríklad v automatických pokladniach a niektorých typoch parkovacích stojanov jedna slovenská spoločnosť využíva počítač s procesorom na taktovacej frekvencii 600MHz a operačnou pamäťou 256MB. Tieto počítače poskytujú vysoký výkon a sú relatívne lacné, pre predstavu sa cena pohybuje okolo 6000 Kč. Počítače sú bez aktívneho chladenia a veľkosť celého počítača bez displeja je 3.5 palcov. Pohodlne sa zmestia aj do menších zariadení. S takouto výpočtovou silou je možné využívať aj softvérové technológie, ktoré boli doteraz nedostupné z dôvodu nedostatku výpočtového výkonu. Medzi inými, aj technológie webových služieb.

Vráťme sa k diskutovanému problému. Ak by existoval softvér, ktorý spĺňa legislatívne požiadavky na evidenciu daňových dokladov a bol by prístupný napríklad cez webovú službu, potom samotné vytvorenie, archivácia a kalkulácie nad daňovými dokladmi sa dajú elegantne presunúť na takýto softvér. Napríklad automatická pokladňa by mohla vyvolať webovú službu evidencie daňového dokladu. Táto služba zaúčtuje položky a potvrdí pokladni úspešnosť operácie. Pokladňa na základe potvrdenia o zaúčtovaní vydá zákazníkovi daňový doklad. Ďalšie spracovanie daňových dokladov a podkladov pre finančný úrad je už plne v kompetencii odlišného softvéru, ktorý preberá túto funkciu. Z pohľadu obchodnej logiky sa jedná vlastne o pozíciu účtovníka, ktorý na požiadanie vystavuje daňové doklady a eviduje ich pre potreby firemného účtovníctva a daňového úradu.

Ako bolo spomenuté, v každej krajine sú spravidla odlišné normy na tvorbu daňových dokladov. Splnenie týchto noriem môže byť delegované na program lokálneho dodávateľa , ktorý je vytvorený špeciálne podľa lokálnych noriem. Vďaka možnosti integrácie je tvorca parkovacieho systému odbremenený od implementácie účtovného systému. Naviac sa týmto spôsobom dá parkovací systém relatívne jednoducho integrovať do účtovného systému majiteľa parkoviska, čo môže byť veľmi zaujímavé.

Uvedieme ďalší príklad. Nech je parkovací systém vlastnený napríklad diskontnou predajňou potravín. Takáto predajňa používa svoj vlastný systém, ktorý eviduje všetky

predané položky. Naviac tento informačný systém vytvára výstupy pre manažment o rôznych aspektoch predaja. Z pohľadu manažmentu, ak chcú monitorovať aj parkovací systém, musia sa naučiť ako zachádzať s jeho obslužným programovým vybavením tak, aby vydolovali žiadané informácie. Naviac je pravdepodobné, že manažment bude vyžadovať takú grafickú úpravu výstupov, ktorú nie je možné jednoduchým spôsobom do parkovacieho systému pridať. Ak by sa však parkovné predávalo, z pohľadu pokladničného systému diskontnej predajne, ako služba, nemusí sa manažment učiť ovládať softvér parkoviska. Účtovné informácie o chode systému získa z informačného systému predajne. Manažéri naviac údaje získajú z pre nich známeho rozhrania. Samozrejme, že v tomto prípade by sa jednalo o podmnožinu údajov, ktoré dokáže poskytnúť parkovací systém. Ak by však tieto boli dostatočné, a je predpoklad že by boli, tak popísanou integráciou získajú oba systémy. Parkovací systém nemusí udržovať účtovné dáta a manažment predajne získa lepší prehľad o celkovej ekonomickej aktivite prevádzky. Výhodou je aj, že v prípade predajní sa doklady potrebné pre evidenciu tržieb spravidla spravujú centrálne. Daňové uzávierky sa tlačia na jednom termináli pre všetky pokladne. Naviac v prípade diskontného reťazca informačný systém predajne automaticky poskytuje finančné výkazy do centrály reťazca a nadriadeným osobám. V prípade integrácie sú aj finančné aspekty prevádzky parkoviska obsiahnuté vo finančných výkazoch predajne a nie je potrebné vytvárať ďalšie komunikačné kanály smerom od predajne k vedeniu.

Autor práce popísané riešenie ponúkol diskontnej predajni a odozva bola pozitívna. V súčasnosti prebieha analýza možností prepojenia na strane informačného systému diskontnej predajne.

Zhrnieme si diskutované body: použitie webových služieb je technicky možné prevádzkovať aj na jednoduchších jednoúčelových zariadeniach. Ukázali sme dva príklady, kde sa využitie integrácie systémov ukazuje ako prínosné, konkrétne pri riešení lokalizačných problémov, alebo pri eliminovaní duplicitných činností dvoch a viacerých systémov.

5.2.2 Možnosti integrácie bez využitia webových technológií

Pri diskutovaných príkladoch je zrejmý prínos integrácie. Ale integráciu je možné docieliť nielen s použitím normy SOAP a webových služieb. Výhody, ktoré norma SOAP prináša sú hlavne z oblasti štandardov. Je veľmi pravdepodobné, že v prípade využitia tejto normy v oboch spájaných systémoch bude integrácia možná aj bez nutnosti významných zásahov do programového vybavenia jednotlivých systémov. Pri integrácii je možné využiť

celú paletu nástrojov, ktoré významným spôsobom integráciu zjednodušia a pomôžu pri odstránení chýb systému. Naviac, ak sa pri tvorbe systému použije vhodný SOAP toolkit, ako napríklad gSOAP v prípade C++ a je dobrý predpoklad, že tento toolkit sa bude aj naďalej vyvíjať, získa tvorca programu v budúcnosti mnoho z funkcií práve vytváraných špecifikácií WS-*. V tomto prípade bude možné jednoducho zmenou verzie knižnice toolkitu a správnym nastavením parametrov takmer bezpracne získať napríklad podporu zabezpečenia SOAP správ prostredníctvom WS-Security. Naviac ak aj druhá strana podporuje, alebo v budúcnosti zamýšľa podporovať tieto štandardy, tak ich nasadenie bude vyžadovať menej úsilia na oboch stranách.

Na druhú stranu, ak by sa na prepojenie použila niektorá neštandardná možnosť, ako napríklad posielanie samotného XML dokumentu cez HTTP, tak je možné, že výsledné riešenie bude implementované rýchlejšie. Z veľkej časti odpadne zoznamovanie sa so štandardami a na jednoduchú komunikáciu je takéto riešenie dostatočné. Naviac je pravdepodobné, že bude vyžadovať menej výpočtového výkonu a tým pádom bude vhodnejšie aj pre nasadenie do jednoúčelových zariadení. Avšak týmto rozhodnutím stráca systém perspektívu jednoduchej integrácie s ostatnými systémami alebo začlenenie do celkového informačného systému zákazníka. Je zrejmé, že pri prvotnej integrácii nebude problém, aby sa dve strany dohodli na formáte dát. Problém môže nastať pri ďalšej integrácii systému, ak zistíme že druhá strana preferuje odlišnú integračnú platformu a naviac ju má aj implementovanú. Vytvorené riešenie s ňou nebude spolupracovať a pre druhú stranu môže byť finančne a časovo náročné prispôsobiť sa neštandardnému riešeniu. Rovnaké tvrdenie platí aj v opačnom smere. To znamená, že architekti systému musia do veľkej miery predvídať smerovanie technológií a musia dokázať určiť ktoré riešenie bude viac preferované v konkrétnom prostredí. Autor práce sa domnieva, že práve SOAP má potenciál stať sa takýmto univerzálne prijatým riešením. A to hlavne z dôvodu štandardizácie a masívnej podpory veľkých softvérových spoločností. V neposlednom rade sú SOAP a webové služby „marketingová nálepka" a do určitej miery môžu napomôcť predaju takto označených systémov.

5.2.3 Príklad integrácie bez využitia normy SOAP

V tejto kapitole popíšeme jeden konkrétny príklad integrácie dvoch systémov. Jedná sa o v praxi nasadené a používané integračné riešenie, ktoré vytvorili dve stredne veľké spoločnosti s rozdielnym technologickým zázemím. Z dôvodu zachovania obchodného

tajomstva nie je možné uviesť mená týchto spoločností, a preto si ich označíme ako spoločnosť Soft a spoločnosť Hard.

5.2.3.1 Predstavenie spoločností

Spoločnosť Soft má vyvinutý vlastný softvérový systém pre riadenie hotelových a reštauračných prevádzok. V zázemí spoločnosti je kvalifikovaný tím programátorov, ktorý pod vedením skúseného hlavného programátora vyvinul úspešný systém. Zároveň je tím pripravený tento systém rozširovať a uspokojovať aj neštandardné požiadavky zákazníka.

Spoločnosť Hard ponúka svojim zákazníkom riešenia z oblasti hardvéru. Jedná sa o turnikety, brány a závory vlastnej výroby. V rámci skvalitňovania a rozširovania svojich služieb firma Hard ponúka k svojim výrobkom aj programové vybavenie, ktoré v spojení s bezdotykovou technológiou čipových kariet predstavuje spoľahlivý prístupový systém. Spoločnosť nezamestnáva programátorský tím, ale zákazky na programové vybavenie realizuje pomocou dodávateľov. Tieto zákazky sú riadené zodpovednou osobou – hlavným programátorom. V náplni jeho práce je zabezpečenie podpory pre súčasné programové vybavenie, dopĺňanie jeho funkcií vlastnými silami a dohliadanie na prácu dodávateľov programových riešení.

Ako vidíme z popisu spoločností, ich know-how v oblasti softvérových riešení je diametrálne odlišné. Zatiaľ čo činnosť firmy Soft je bytostne závislá na vytváraní programových riešení, ťažisko činnosti spoločnosti Hard tkvie v predaji turniketov. Avšak aj firma Hard je schopná dodať špecializovaný softvér, ktorý ovláda ich hardvér.

5.2.3.2 Základné požiadavky na systém

Tieto dve spoločnosti sa stretli ako dodávatelia na projekte zábavného centra s hotelom, bazénmi, saunami a ďalšími atrakciami. Spoločnosť Hard sa zaviazala dodať prístupový systém do areálu, zatiaľ čo spoločnosť Soft je dodávateľom hotelového a reštauračného riešenia. Vzhľadom k tomu, že prístupový systém riadi aj vstup do hotela a atrakcií vznikla potreba oba systémy integrovať. Prístup do hotelovej časti, ako sa ukázalo, nebol jediným styčným bodom oboch systémov. Predstava majiteľa zábavného centra zahŕňala aj rozdelenie centra do rôznych zón. Pohyb medzi zónami musí byť riadený a vstup do jednotlivých zón musí byť určený prístupovými právami. Tieto práva sa stanovia na základe typu zakúpenej vstupenky. Naviac v centre existujú aj služby, ktoré sú spoplatnené na

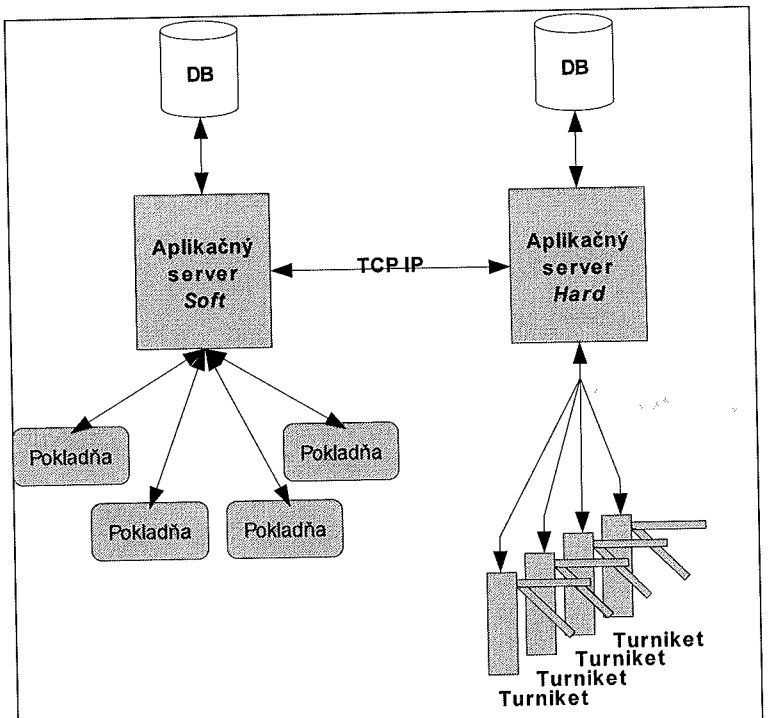
základe ich využitia. Medzi tieto služby patrí jednorázový vstup do sauny, masáž a podobne. Z pohľadu majiteľa, systém musel zabezpečovať bezhotovostné účtovanie poskytnutých služieb, ktoré zákazník zaplatí pred odchodom z centra. Keďže v areáli sú bazény a iné mokré atrakcie, muselo sa v celom areáli eliminovať použitie peňazí. Majiteľ presadzoval použitie elektronickej peňaženky aj z dôvodu eliminovania krádeží v areáli. Každý zákazník pri vstupe do areálu obdrží bezkontaktnú čipovú kartu vo forme náramku, ktorou sa identifikuje pri prechodoch cez turnikety, pri pridelení šatňovej skrinky, ako aj pri objednávaní rôznych služieb, alebo pri konzumácii v reštaurácii, či bare. Z týchto požiadaviek je zrejmé, že systémy oboch spoločnosti musia komunikovať vo viacerých rovinách. Jednak systém turniketov musí na základe oprávnení umožňovať prístup do zón. Naviac sa však cez turniketový systém realizuje aj „predaj" služieb akými sú vstup do sauny a podobne. Podobne v reštaurácii musia byť zákazníci schopní „platiť" pomocou bezdotykovej karty. Oba systémy musia naviac rešpektovať zvolené maximum čerpania kreditu na kartu a nesmú umožniť prečerpanie tohto limitu. Táto podmienka slúži ako poistka v prípade straty identifikačnej karty. Limit je v praxi nastavený v rovnakej výške ako zmluvná sankcia za stratu identifikačného čipu. Týmto obmedzením sa zamedzuje zneužitiu karty zo strany zákazníka.

Zo zadania vyplýva že oba systémy musia intenzívne spolupracovať a vymieňať si informácie a z pohľadu majiteľa sa oba systémy musia správať ako keby boli jedným systémom.

5.2.3.3 Zvolené riešenie

Ako bolo zmienené v predstavení spoločností, je len prirodzené, že návrhu riešenia sa zhostila firma Soft. Počas spoločnej komunikácie sa obe spoločnosti dohodli na riešení, ktorého implementácia bola v silách oboch partnerov. Na zvolené riešenie nemalým spôsobom vplýval aj skorý termín odovzdania systému do prevádzky, rádovo dva mesiace od úvodného stretnutia. Z nedodržania termínu by pre obe spoločnosti vyplývali nemalé zmluvné sankcie. Zo skúseností z predchádzajúcich projektov sa snažila firma Soft navrhnúť riešenie, ktoré by bolo dostatočne univerzálne a do budúcnosti by zabezpečilo možnosť prepojenia aj s prístupovými systémami iných dodávateľov. Spoločnosť Soft má veľkú časť svojho systému implementovanú s využitím jazyka Java a programové vybavenie turniketov je napísané v jazyku C++. Prvotný návrh prepojenia systémov mal byť realizovaný na platforme JMS (*Java Message System*). Táto technológia bola zamietnutá spoločnosťou Hard, lebo jej produkty sú vyvíjané na platforme C++. Obe spoločnosti hľadali rozumné riešenie na základe technológií,

ktoré im boli známe. Spoločne sa dohodli na použití prenosu cez TCP-IP so správami v tvare XML. Tieto technológie boli dostatočne známe pre obe spoločnosti a zodpovední pracovníci dokázali rýchlo odhadnúť čas potrebný na implementáciu daného protokolu z ich strany.



Obrázok 1: Schéma prepojenia aplikácií

Výsledkom spoločných rokovaní bol pomerne obsiahly dokument, ktorý presne popisuje zvolený protokol a štruktúru prepojenia systémov. Dokument deklaruje snahu o maximálne možné oddelenie oboch systémov tak, aby ich komunikácia a prienik bola realizovaná výhradne za použitia protokolu TCP-IP. Jednotlivé pokladne sú pripojené k aplikačnému serveru Soft, pričom každá z nich má na serveri vyhradené samostatné vlákno. Aplikačný server Soft pre každú pokladňu vytvorí samostatné spojenie s aplikačným serverom Hard. Jednotlivé komunikačné kanále sú na sebe nezávislé. Komunikácia je naviac navrhnutá ako bezstavová a prebieha spôsobom otázka odpoveď. Na každú otázku musí prísť odpoveď. Jednotlivé správy sú komprimované s využitím metódy ZIP.

Komunikácia je rozdelená na správy, kde je každá správa vložená do komunikačného rámca. Rámec pozostáva z hlavičky, dĺžky posielaných dát, dĺžky nekomprimovaných dát, CRC a nasledujú samotné komprimované dáta.

Telo správy je po rozbalení tvorené XML dokumentom s presne definovanou štruktúrou. Na kódovanie XML správ bola zvolená kódová stránka cp1250. Komunikácia je v zásade vyvolávaná z aplikačného servera Soft. Aplikačný server Hard spracuje požiadavku a

vráti odpoveď, alebo výnimku.

```
<?xml version="1.0" encoding="WINDOWS-1250"?>
<protokol verzia="1" znacka="1" sprava="123">
      <prikaz meno="test">
            <hodnoty>
                  <hodnota meno="echo" typ="string">Testovanie</hodnota>
            </hodnoty>
      </prikaz>
</protokol>
```

Text 2: Príklad výzvy

V texte je zapísaný príklad výzvy. Všimneme si, že v riešení je použité číslovanie správ.

```
<?xml version="1.0" encoding="WINDOWS-1250"?>
<protokol verzia="1" znacka="1" sprava="123">
      <navrat>
            <hodnoty>
                  <hodnota meno="echo" typ="string">Testovanie</hodnota>
            </hodnoty>
      </navrat>
</protokol>
```

Text 3: Príklad odpovede

V texte je uvedený príklad korektnej odpovede od systému. Takáto odpoveď implikuje správne spracovanie výzvy. Navrhnutý systém obsahuje ešte aj jednoduché potvrdenie prijatia a spracovania správy bez dodatočnej informácie.

```
<?xml version="1.0" encoding="WINDOWS-1250"?>
<protokol verzia="1" znacka="1" sprava="123">
      <vynimka>
            <text>Textovy popis vynimky</text>
            <debug>
                  <!-- popis chyby pre odladenie - zapíše sa do logu -->
            </debug>
      </vynimka>
</protokol>
```

Text 4: Príklad výnimky

V prípade, ak požiadavka z nejakého dôvodu nie je vykonaná korektne, je prijímací systém povinný o tejto skutočnosti upovedomiť odosielateľa vygenerovaním XML správy. V tejto XML správe je zahrnutý popis pre obsluhu a popis pre nájdenie prípadnej chyby, ktorý sa užívateľovi nezobrazí, ale uloží sa do logu aplikácie.

V ďalšej diskusii medzi spoločnosťami Soft a Hard už dominovali výhradne praktické

záležitosti. Boli definované rôzne typy správ s ich odpoveďami a zosúladila sa logika postupov pri zaradení karty do systému, pri jej fakturácii pred opustením areálu a ďalšie podrobnosti zahŕňajúce získavanie relevantných informácií pre reklamácie zákazníkov.

5.2.3.4 Analýza zvoleného riešenia

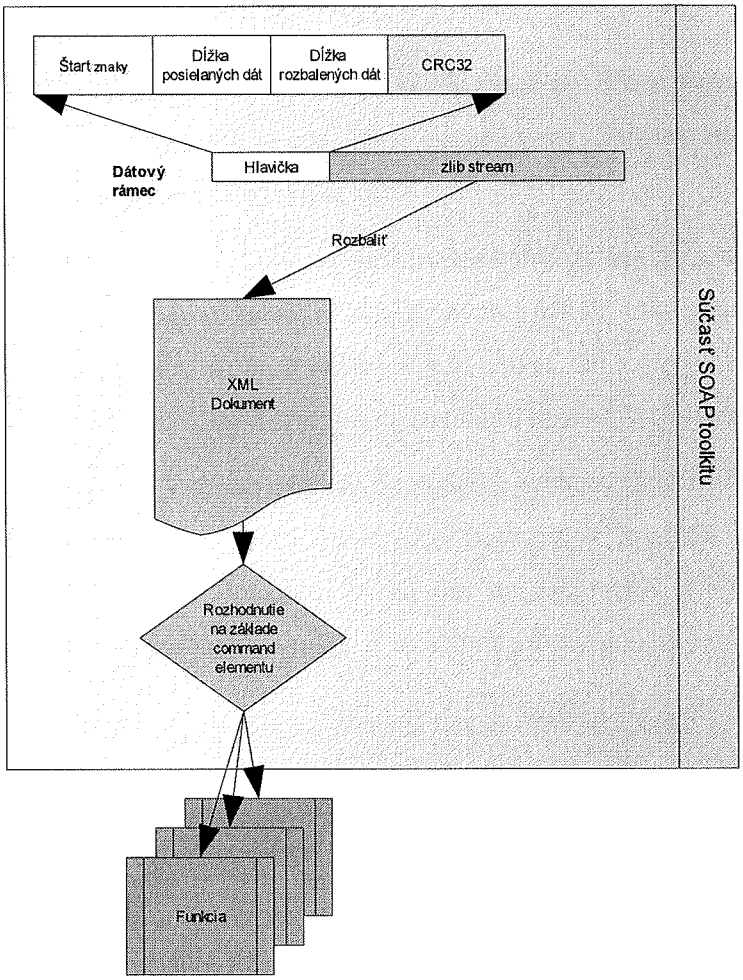
Popísané riešenie integrácie systémov v praxi ukázalo, že zvolený prístup bol realizovateľný. Systém bol odovzdaný do užívania v zmluvne stanovenom termíne. V tejto kapitole poukážeme na silné a slabé stránky návrhu.

Na prvý pohľad zvolené riešenie vykazuje výraznú podobnosť so systémami založenými na webových službách. Podobnosť s protokolom SOAP je nápadná. Prvé nie úplne dobré miesto návrhu je už samotná definícia hlavičky protokolu. Konkrétne sa v hlavičke protokolu definuje veľkosť prenesených dát, veľkosť dát po rozbalení. Oba údaje sú minimálne duplicitné. Konzistencia prenesených dát je zabezpečená prenosovým protokolom TCP. Veľkosť dát po rozbalení je obsiahnutá v hlavičke zip streamu. Preto ani jeden z týchto parametrov nie sú nutné. Použitie CRC na zisťovanie konzistencie správy tiež zrejme nie je úplne na mieste, pre samotný obsah prenášaných dát je už zahrnuté v zip streame a samotná hlavička nie je obsiahnutá v tejto kontrolnej sume, takže prínos je nulový. Korektný prenos správy je takisto garantovaný použitím protokolu TCP-IP. Takže vo výslednom protokole je CRC dátového obsahu zahrnuté minimálne tri krát. Zaujímavé je aj špecifikovanie začiatku správy dopredu zvoleným reťazcom. Samozrejme nie je zaručené, že sa tento nebude vyskytovať v samotnej správe. Autor sa domnieva, že by bolo rozumnejšie, pri použití komprimovaného streamu použiť vlastnosti tohto streamu. Všetky informácie obsiahnuté v hlavičke paketu sa nachádzajú aj v hlavičke streamu a vytváranie duplicitnej hlavičky príde autorovi práce zbytočné.

Odhliadnuc od hlavičky správy, zamerajme sa na jej obsah – samotné dáta. Tieto sú komprimované pomocou zlib algoritmu. Obe strany musia implementovať kompresný a dekompresný algoritmus. Alebo tento prevziať z knižnice. Ďalej obe strany musia implementovať mechanizmus na kontrolu čísla správy, ako aj rutinu na desierializáciu prijatých dát. V neposlednom rade je potrebné implementovať aj mechanizmus na spustenie správnej funkcie podľa kontextu prijatého XML dokumentu.

5.2.3.5 Alternatívne riešenia

Prvým naivným riešením tohto konkrétneho problému integrácie dvoch systémov by bolo vytvorenie proprietárneho protokolu. Tento protokol by nemusel byť postavený na báze XML, ale postačoval by binárny protokol podporujúci RPC. Protokol by mohol vyzerať napríklad tak, že by sa určil formát jednotlivých správ pre konkrétne funkcie, podobne ako je to bežné pri komunikácii dvoch zariadení prostredníctvom sériovej linky. Je vidieť, že autori výsledného riešenia majú skúsenosti s prepojovaním systémov, lebo takémuto naivnému riešeniu sa vyhli. Celkom vhodne zvolili jazyk XML ako základ rozšíriteľného protokolu. Nad týmto jazykom popísali systém kódovania jednotlivých typov parametrov, ako sú integer, alebo reťazec. Autori zrejme mali určitú predstavu o existencii webových služieb, ktoré do určitej miery napodobnili. Dá sa povedať, že zo špecifikácie webových služieb vybrali niektoré princípy a vytvorili svoj vlastný proprietárny protokol. V tomto riešení bol kladený dôraz na rýchlosť implementácie. Zvolený postup, pri ktorom bol od základu vybudovaný celý protokol, bol však časovo náročný jednak na návrh ako aj na implementáciu.



Obrázok 2: Spracovanie správy

Všetky horeuvedené vlastnosti navrhnutého riešenia sú zahrnuté v norme SOAP, a existujú nástroje a knižnice, ktoré výrazným spôsobom prispievajú k zrýchleniu vývoja. Na obrázku 2 je znázornený proces prijatia správy. Funkčnosť zahrnutá priamo, alebo nepriamo v norme SOAP je zvýraznená. Komprimácia dát je možná pri použití http protokolu. Číslovanie správ je zahrnuté vo WS-Reliable Messaging[43]. Overenie prijatého XML dokumentu podľa XSD je tiež vlastnosťou SOAP. Mechanizmus na spustenie funkcie býva obvykle súčasťou SOAP toolkitu, ktorý na základe wsdl vygeneruje prázdne implementácie pre dané funkcie.

Ak by sa namiesto vytvorenia celého protokolu od základu použila napríklad norma SOAP, museli by analytici aj programátori stráviť čas štúdiom normy. Ako sa ukazuje takto strávený čas by sa v konečnom dôsledku vyplatil. Namiesto špecifikovania a implmentovania úplne nového protokolu by sa použilo štandardizované riešenie. Naviac je zrejmé, že norma SOAP a pridružené špecifikácie dnes poskytujú všetky požadované vlastnosti.

5.2.3.6 Zhrnutie

Ako je vidieť autori riešenia integrácie sa do značnej miery priblížili známym a používaným praktikám. Ich riešenie je ale úplne proprietárne. Prínosy štandardizovaných riešení sú zjavné a preto ich nebudeme rozpisovať.

Aj keď sa dá oprávnene pochybovať o univerzálnosti riešenia aj keby boli použité štandardy ako SOAP, je zrejmé, že použitie SOAP urýchli čas na implementáciu. Takisto v oblasti návrhu je výhodné poučiť sa zo štandardov a využívať ich v najväčšej možnej miere. Naviac, ako už bolo zmienené, sú výhody použitia štandardizovaného riešenia nesporné.

Autor sa domnieva, že v tomto prípade by použitie webových služieb bolo na mieste. V prípade nasadenia podobného riešenia, kde by bol zúčastnený len jeden z dodávateľov a druhý by bol nahradený iným subjektom, by sa čas na prepojenie systémov pri použití webových služieb výrazným spôsobom znížil. A to hlavne vďaka použitiu dostupných knižníc SOAP.

Takisto by sa tieto systémy mohli lepšie začleniť do prípadného celkového informačného systému spoločnosti, ak by tento využíval webové služby.

5.2.4 Príklad integrácie s využitím normy SOAP

V predchádzajúcej kapitole sme sa zoznámili s riešením, ktoré pri integrácii nevyužilo výhody webových služieb. V tejto kapitole si priblížime prípad, ktorý je zaujímavý z obchodného hľadiska a využíva normu SOAP. V popise sa zameriame len na aspekty systému, ktoré sú potrebné pre pochopenie navrhnutého riešenia s využitím normy SOAP.

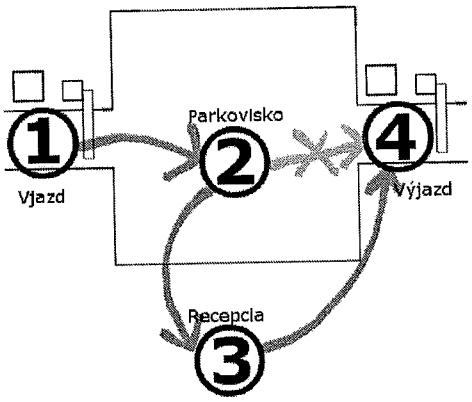
5.2.4.1 Počiatočný stav

V polovici roku 2006 bola firma v ktorej autor pracuje, nazvime si ju Park, vybraná ako dodávateľ prístupového systému do administratívneho komplexu. Jeho súčasťou je aj parkovacia plocha. Vjazd na ňu mal byť riadený systémom. V ponuke firmy Park bol preto zahrnutý plnohodnotný parkovací systém vlastnej výroby, a prístupový systém Granta do budovy od firmy Siemens. Investor nebol s ponúkaným riešením spokojný. Podľa jeho predstavy malo byť parkovisko vyhradené čisto pre zamestnancov firiem sídliacich v budove a preto sa mu zdalo neefektívne kupovať si plnohodnotný parkovací systém. Investor nebral do úvahy prípadných návštevníkov daných firiem, ktorí nemajú vydané prístupové karty a

tým pádom nemôžu parkovisko využívať. Naviac využitie časti parkoviska pre náhodných parkujúcich generuje určitý príjem.

V rámci zníženia nákladov bolo rozhodnuté, že funkcie parkovacieho systému budú nahradené za pomoci prístupového systému Granta. Toto nahradenie bolo možné z dôvodu, že investor neplánoval spoplatnenie parkoviska pre náhodných parkujúcich. Napriek tomu, sa firme Park podarilo presadiť, aby bol za pomoci systému Granta na parkovisko vytvorený návštevnícky systém.

Tento systém pracuje nasledovne: Pri vjazde na parkovisko si návštevník vyžiada návštevnícku kartu, ktorú mu vjazdový stojan automaticky vydá. Počas vydávania karty sa táto skutočnosť v prístupovom systéme Granta javí ako prechod danej karty cez prístupový bod, čím sa karta dostane do zóny jedna. Následne je návštevník povinný si na recepcii kartu označiť na čítacom zariadení, čím sa pre systém Granta dostane karta do zóny dva z ktorej je možné potom uskutočniť prechod cez výjazdový stojan do zóny mimo parkoviska. Výjazdový stojan návštevnícku kartu zadrží. Tento princíp sa podarilo pomocou programových vlastností systému Granta naprogramovať, sprevádzkovať a celý systém odovzdať investorovi.



Obrázok 3.: Činnosť systému

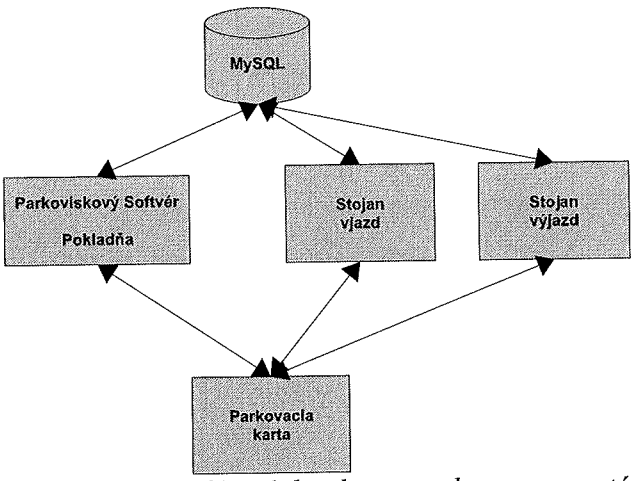
Obrázok ukazuje činnosť systému. V bode 1 je návštevníkovi vydaná karta vstupným stojanom. Následne návštevník zaparkuje, čo je vyznačené bodom 2. Parkovisko nemôže opustiť do doby, kým si na recepcii neoznačí parkovaciu kartu. Súčasťou označenia karty je zaznamenanie údajov o firme v rámci budovy, ktorú návštevník navštívil. Po označení karty na recepcii je mu umožnený výjazd z parkoviska, kde mu kartu automaticky odoberie výstupný stojan, na obrázku označené bodom štyri.

Po zhruba mesiaci prevádzky tohto systému investor zmenil názor a uvedomil si, že je pre neho nevýhodné, ak nedokáže parkovanie návštev riadne spoplatniť. Preto sa rozhodol, že bude po spoločnosti Park požadovať rozšírenie systému o možnosť platby. Z pohľadu firmy

Park by to znamenalo vymeniť hardvérové časti vjazdového výjazdového stojana a nahradiť ich časťami z parkovacieho systému. V podstate by sa jednalo o pôvodné navrhované riešenie. Lenže pre investora takáto investícia neprichádzala do úvahy. Zrejme z dôvodu, že by jeho vedúci pracovníci museli priznať, že pri špecifikácii požiadaviek ignorovali doporučenia a že na základe ich chybných rozhodnutí zakúpili časť hardvéru, ktorá je nie je vhodná a musí byť vymenená, čím spôsobili finančné straty. Tieto straty sa netýkajú čisto samotnej výmeny hardvéru stojanov, ale aj celej kabeláže parkoviska. Zmena kabeláže, sama o sebe, bola vyčíslená rádovo v desiatkach tisícov korún.

5.2.4.2 Navrhnuté riešenie

Vzhľadom na zmienené faktory sa firma Park pokúsila vytvoriť riešenie s využitím hardvéru, ktorý už investor vlastní. V podstate sa jedná o integráciu parkovacieho softvéru s hardvérom prístupového systému Granta. Prístupový systém musí z pohľadu parkoviska simulovať činnosť parkovacích stojanov v plnej miere.



Obrázok 4: Schéma dátových tokov v parkovacom systéme

Princíp využívaný v parkovacom systéme je znázornený na obrázku. Dôležitý aspekt parkovacieho systému je, že dáta potrebné na rozhodovanie prechodových bodov a stojanov sú zapísané priamo v pamäti karty. Týmto spôsobom firma Park zabezpečuje základné funkcie systému aj v prípadoch, kedy je centrálna databáza nedostupná, lebo sa prechodové body dokážu rozhodovať na základe dát prečítaných z karty. Rovnako aj pokladne parkovacieho systému údaje o dobe parkovania čerpajú priamo z parkovacej karty a po úhrade parkovného ich patričným spôsobom pozmenia.

Tejto vlastnosti parkovacieho systému využila firma Park pri špecifikovaní riešenia a

navrhla systém, v ktorom sa do parkoviskového softvéru pridá softvérový modul čítačky kariet, ktorý namiesto prístupu k internej pamäti karty bude údaje o karte čítať z prístupového systému. Zápis údajov na kartu, podľa ktorých potom parkovacie stojany rozhodujú o povolení prechodu, bude riešený predaním týchto informácií do prístupového systému a zabezpečením korektnej funkcionality tohto systému vzhľadom k zákazníkovi.

Prístupový systém Granta pracuje s databázou Microsoft SQL. V nej má uložené všetky potrebné údaje na povolenie, alebo zamietnutie prechodu niektorej karty cez prístupový bod. Zmenou týchto údajov sa dá docieliť správna funkcia prístupového systému.

Pracovník firmy Park, ktorý je špecialistom na tento prístupový systém je schopný vyvíjať softvér len s využitím Visual Basic, čím sa možné prepojenie systémov parkoviska a prístupu skomplikovalo. Zároveň bolo nutné, aby prepojenie z parkovacieho softvéru, ktorý nie je tenkým klientom, bolo realizovateľné z ktoréhokoľvek počítača umiestneného na lokálnej sieti. Na splnenie týchto požiadaviek sa ukázalo ako najvhodnejšie riešenie s využitím normy SOAP.

Výsledné navrhnuté riešenie vyzerá nasledovne: Všetok hardvér aj inštalácie zostanú nedotknuté. K prístupovému systému sa naprogramuje webová služba, ktorá bude parkovaciemu softvéru dodávať informácie, ktoré by za štandardných podmienok boli získané čítaním priamo z pamäte parkovacej karty. Namiesto zápisu údajov na kartu, sa tieto odošlú webovej službe, ktorá zmenou údajov pre kartu v databázi prístupového systému zabezpečí korektné správanie sa tohto systému smerom k zákazníkovi. Ďalej sa údaje o prechodoch kariet cez parkovacie stojany z prístupového systému na vyžiadanie posielajú do parkovacieho systému, čím sa zabezpečí kompletné prepojenie oboch systémov.

Parkovací systém bude mať k dispozícii všetky údaje, rovnako ako keby sa jednalo o štandardné parkovacie stojany. Tým pádom budú k dispozícii aj všetky štatistické zostavy a parkovací systém bude pracovať štandardným spôsobom, ako keby boli k nemu pripojené natívne parkovacie stojany. Existuje drobná výnimka v tomto tvrdení, a síce nebude k dispozícii monitoring stojanov, ale tie sú monitorované cez softvér prístupového systému.

5.2.4.3 Implementácia

Navrhované riešenie, ako už bolo zmienené, je postavené na norme SOAP. Jedná sa veľmi jednoduché nasadenie, kde webová služba poskytuje tri základné volania – ReadCard, WriteCard a GetPassage, ktoré sú z princípu RPC. RPC sú preto, lebo sa vyžaduje okamžitá

odozva zo strany webovej služby. Presný popis rozhrania nie je pre túto prácu dôležitý a preto ho nebudeme popisovať. Webový server je IIS s implementovanou službou vo Visual Basicu. Parkovací systém je implementovaný v Borland C++ Builder. V jeho prípade autor použil open source knižnicu gSOAP. Implementácia prepojenia prebiehala vcelku hladko, až na už zmienený problém s prenosom dátového typu xsd::dateTime. Tento problém bol vyriešený tak, že sa prenáša dátum a čas zakódovaný do desatinného čísla. Tento spôsob kódovania je typický pre OLE automation. Implementácia oboch častí prepojenia, klienta aj servera trvala autorovi, ktorý nie je oboznámený s programovaním vo Visual Basic, 4 hodiny. Väčšia časť tohto času bola pritom venovaná korektnému prenosu dátumu a času. Po vytvorení kostry prepojenia budú implementované časti, ktoré zabezpečujú funkčnosť a nie sú zaujímavé pre túto prácu.

5.2.4.4 Záver

Ako bolo ukázané využitie časti funkcionality webových služieb výrazným spôsobom urýchlilo a zlacnilo implementáciu prepojenia dvoch systémov, šitého na mieru zákazníkovi. V porovnaní s predchádzajúcim príkladom je vidieť výraznú úsporu času a prostriedkov. Aj keď je tento druhý príklad jednoduchší, je asi len ťažko predstaviteľné, že by jeden človek bol schopný implementovať protokol prepojenia z predchádzajúceho príkladu v tak krátkom čase a naviac bez podrobných znalostí jedného z programovacích jazykov.

Ako bolo spomenuté v úvode kapitoly, je tento príklad zaujímavý hlavne z obchodného hľadiska. Vďaka využitiu normy SOAP bola spoločnosť Park schopná v krátkom čase a s minimalizovaním nákladov dodať riešenie, ktoré uspokojilo významného klienta.

V dnešnej dobe sa autor čoraz častejšie stretáva s požiadavkami zákazníkov na prepojenie parkoviskového systému s ostatnými systémami zákazníka. Ako príklad môžeme uviesť Národné Tenisové Centrum v Bratislave, kde sa bude realizovať prepojenie s vernostným systémom firmy Smarcom. Aj toto prepojene bude realizované s využitím normy SOAP. Takisto ako aj prepojenie s firmou Siemens za účelom úhrady parkovného cez systém mobil parking a mnohé ďalšie.

5.3 Výhody webových služieb

Webové služby ponúkajú jednoduchú a priamočiaru cestu k integrácii rôznorodých systémov naprieč rôznymi platformami programových jazykov ako aj operačných systémov.

Ponúkajú štandardizované riešenie podporované významnými tvorcami programových nástrojov. Je nespornou výhodou z pohľadu dodávateľa systému ako aj nákupcu, ak je systém pripravený na prepojenie s ďalšími systémami. Takúto pripravenosť je potrebné riešiť komplexne na viacerých úrovniach od návrhu systému cez správny výber integračnej technológie.

Použitie technológií webových služieb sa neobmedzuje len na nové aplikácie. V práci boli diskutované techniky, ktoré sa dnes využívajú, na začlenenie stávajúcich systémov do celkového informačného systému spoločnosti. Pomocou týchto techník je možné takéto *legacy* systémy a ich rozhrania naviac aj monitorovať.

Z pohľadu nákupcu systému hrá nemalú strategickú výhodu, ak je jeho informačný systém dobre štruktúrovaný. Možnosť voľného prepojenia viacerých aplikácií do súrodého celku s dobre definovanými rozhraniami ponúka možnosť voľby dodávateľa jednotlivých subsystémov. Týmto spôsobom je pri budovaní informačného systému odbúraná závislosť na jedinom dodávateľovi. Takisto sa zlepšuje možnosť inkrementálneho budovania systému.

Významným prvkom podporujúcim možnosť inkrementálneho budovania systému je v neposlednej rade aj tlak na dobré zarovnanie rozhraní systémov s obchodnými procesmi. Dobre zarovnané rozhrania umožňujú lepšiu orientáciu v systéme pre manažérov, čím zlepšujú ich schopnosť zostaviť celý systém z podsystémov.

Z technickej stránky by medzi výhody webových služieb autor zaradil veľmi dobrú podporu zo strany programovacích nástrojov. Nástroje výrazným spôsobom urýchľujú implementáciu a uľahčujú ladenie programov.

5.4 Nevýhody webových služieb

Medzi najvýraznejšie nevýhody použitia webových služieb by autor zaradil fakt, že sa jedná o pomerne novú technológiu, ktorá nie je overená dlhoročnou praxou. Neustále vznikajú nové a upravujú sa pôvodné štandardy z oblasti webových služieb. Pravdou je, že základné normy webových služieb akými sú SOAP, WSDL, XML schéma sa už dlhšiu dobu nemenia.

Norme SOAP chýbajú niektoré základné vlastnosti, ako je napríklad adresácia. Tieto chýbajúce vlastnosti sa do webových služieb pridávajú cez mechanizmus rozšírení. Je otázne koľko z týchto rozšírení normy sa nakoniec uplatní do takej miery, že budú prispievať k všeobecnej interoperabilite. Zrejme to budú tie, ktoré vo svojich knižniciach na prácu so

SOAP budú poskytovať výrobcovia prevládajúcich nástrojov na tvorbu softvéru.

Z histórie normy CORBA je vidieť, že neefektívny mechanizmus tvorby štandardov môže mať fatálne dôsledky na jeho rozšírenie. Skutočnosť, že rozšírenia normy SOAP sú vyvíjané viacerými štandardizačnými komisiami môže byť predzvesťou vytvorenia duplicitných rozšírení, ktoré nebudú navzájom kompatibilné a budú medzi sebou súperiť o postavenie na trhu. Takýto vývoj by mohol mať neblahé následky na normu SOAP a jej využitie.

5.5 Limitujúce faktory nasadenia webových služieb

Norma SOAP nie je jediným nástrojom na prepájanie systémov. Existujú aj ďalšie praktiky s pomocou ktorých sa za využitia podobných základných technológií dá docieliť rovnaký výsledok. Medzi takéto techniky patrí napríklad prístup REST.

Medzi faktory ovplyvňujúce vhodnosť nasadenia webových služieb založených na norme SOAP by autor zaradil nedostupnosť výpočtového výkonu alebo cenu za prenosové pásmo. Autor sa stretol s prípadom, kedy by z logického hľadiska bolo využitie webových služieb možné, ale cena za sieťové prepojenie, konkrétne GPRS, bola neúmerne vysoká do takej miery, že sa musel zvoliť prístup minimalizujúci počet prenesených dát.

6 Záver

V práci boli popísané viaceré technológie z oblasti webových služieb. Autor sa zameral na využitie týchto technológií v oblasti malých a stredných firiem. Vízia, ktorú webové služby a architektúra SOA vytyčujú sa v praxi presadzuje skoro výhradne len v projektoch veľkých spoločností. Napriek tomu sa niektoré časti webových služieb dajú s úspechom využiť aj v prostredí malých a stredných podnikov. Autor ukázal na príkladoch ako sa výrazne zrýchli návrh a implementácia systému, keď sa využijú niektoré technológie webových služieb. Samozrejme architektúra SOA ponúka vlastnosti a myšlienky, ktoré pri malých projektoch nemá význam využívať. Je zrejmé, že na prepojenie dvoch systémov by použitie orchestrácie webových služieb bolo skôr kontraproduktívne. Avšak len budúcnosť ukáže, či sa nevyskytnú projekty pri ktorých sa využijú aj pokročilé vlastnosti, ktoré webové služby ponúkajú.

Trendy v oblasti využívania IT ukazujú, že riešenia, ktoré boli používané v podmienkach veľkých spoločností sa pomaly presadzujú aj na poli small and medium business (SMB). Preto domnienka, že integrácia systémov aj pri projektoch menších spoločností bude v budúcnosti hrať čoraz výraznejšiu úlohu sa môže ukázať ako pravdivá. Napovedá tomu aj trend s využitím CRM (*Customer relationship management*), alebo SCM (*supply chain management*) systémov na poli SMB. Naviac pri integrácii viacerých systémov sa môže ukázať, že vlastnosti webových služieb sa budú s úspechom používať. V každom prípade, už samotné využitie časti funkcionality webových služieb pri projektoch má za následok oboznámenie sa s týmito technológiami. V prípade potreby vybudovania komplexnejšieho prepojenia viacerých systémov bude na takéto prepojenie potrebné prekonať menšie bariéry v podobe zvládnutia menšieho počtu neznámych technológií. Naviac použitie webových služieb aj na jednoduché prepojenia dvoch systémov ponúka možnosť v budúcnosti začleniť toto prepojenie a jeho rozhrania do celkového firemného IT riešenia.

V práci boli diskutované aj alternatívne prístupy, ktoré sa používajú na jednoduché prepojenie systémov. Niektoré z týchto technológií sú hojne využívané, ako napríklad prístup REST. Autor práce sa domnieva, že aj keď sú tieto technológie postačujúce, neprinášajú taký potenciál budúceho využitia, ako je to v prípade webových služieb, ktoré sú založené na SOAP. Naviac tvorba webových služieb s využitím SOAP je veľmi dobre podporovaná prakticky v ľubovolnom programovacom jazyku a vo väčšine prípadov je priamo zakomponovaná do vývojového prostredia. Takisto túto technológiu výrazne podporujú firmy

ako sú IBM, Microsoft atď.

Diskutované boli aj pravdepodobné dôvody pre minimálne využívanie webových služieb na poli SMB. Tieto zahŕňajú jednak nutnosť prekonať počiatočnú technologickú bariéru zo strany dodávateľov systémov. Ďalším z dôvodov je zrejme aj fakt, že projekty, ktoré vyžadujú integráciu sú vo väčšine prípadov len dva alebo tri. A na takéto prepojenie sa dá použiť aj niektorý z alternatívnych spôsobov. Naviac drvivá väčšina systémov s ktorými sa autor mal možnosť zoznámiť pracuje na platforme operačného systému Windows. Pokiaľ sa pri integrácii neprekračuje hranica operačných systémov, tak sú možnosti prepojenia výrazne bohatšie. Naviac sa často neprekračuje ani hranica jedného počítača v prípade projektov malých a stredných spoločností.

Druhým závažným dôvodom pre ktoré sa webové služby len minimálne používajú v sektore SMB je neexistencia tlaku na ich použitie zo strany zákazníkov. Takýto tlak sa dá očakávať od spoločností, ktorých zodpovední pracovníci majú dobrú predstavu o technologických možnostiach súčasnosti. Takáto znalosť pomerov je zatiaľ skoro výlučne doménou veľkých spoločností alebo spoločností s dobrým technologickým zázemím a vyšším rozpočtom na IT.

Boli zmienené aj nedostatky normy SOAP, ako je tlak zo strany toolkitov na tvorbu webových služieb tak povediac v obrátenom poradí. Namiesto logického postupu, kde sa najskôr vytvorí WSDL dokument, popisujúci rozhrania webovej služby, sa vytvorí implementácia tejto služby a pomocou nástrojov, často obsiahnutých priamo v jazyku, sa strojovo vygeneruje príslušné WSDL. Tento postup je preferovaný hlavne z dvoch dôvodov, a síce že nevyžaduje takmer žiadne znalosti WSDL a fakt, že ručné písanie WSDL je značne zdĺhavé. Tento trend sa pokúša zmeniť návrh novej SOAP vrstvy pre jazyk Java – Alpine.

Celkovo sa javí, že webové služby sa stávajú preferovanou voľbou pre prepájanie aplikácií. Aj v prípade, ak nie sú ich možnosti využité úplne, ale pre konkrétny projekt sa z nich vyberie len časť, ponúkajú zaujímavú alternatívu k bežným spôsobom prepájania aplikácií. Dokonca v niektorých prípadoch, menovite pri prepájaní aplikácií napísaných v jazyku Java a aplikácií vytvorených na platforme Microsoft, sú webové služby jedným z mála schodných riešení prepojenia.

Referencie

[1] Web Services Architecture, 2004,
http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[2] Reference Model for Service OrientedArchitecture 1.0, 2006,
http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[3] Extensible Markup Language (XML) 1.1 (Second Edition),
http://www.w3.org/TR/2006/REC-xml11-20060816/

[4] XML Schema Part 1: Structures Second Edition, 2004,
http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[5] Service-oriented architecture definition,
http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

[6] Reference Model for Service Oriented Architecture 1.0, 2006,
http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[7] W3C XML Protocol Working Group SOAP Version 1.2 Part 0: Primer, 2003,
http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[8] Basic Profile 1.1, 2006,
http://www.ws-i.org/Profiles/BasicProfile-1.1.html

[9] SOAP with Attachments, 2000,
http://www.w3.org/TR/SOAP-attachments

[10] Oasis UDDI,
http://www.uddi.org/

[11] Web Services Description Language (WSDL) 1.1,
http://www.w3.org/TR/wsdl

[12] Box D. A Brief History of SOAP, 2001,
http://webservices.xml.com/pub/a/ws/2001/04/04/soap.html

[13] Java CoreWeb Services,
http://java.sun.com/webservices/technologies/core/index.jsp

[14] Visual Component Library,
ftp://ftpc.borland.com/pub/delphi/devsupport/general/delphi1/vclref.zip

[15] MSXML 4.0 Service Pack 2,
http://www.microsoft.com/downloads/details.aspx?FamilyID=3144b72b-b4f2-46da-b4b6-c5d7485f2b42&DisplayLang=en

[16] Babet B. Unofficial SOAP Bug Fixes, 2002,
http://dn.codegear.com/article/28514
[17] gSOAP C/C++ Web Services and Clients,
http://www.cs.fsu.edu/~engelen/soap.html
[18] Goodin, D., Zabezpeceni webovych sluzeb, 2007
[19] SOAP Web Services Attacks,
http://www.spidynamics.com/assets/documents/SOAP_Web_Security.pdf
[20] Schneier B. Crypto-Gram News Letter SOAP, 2000,
http://www.schneier.com/crypto-gram-0006.html#SOAP
[21] Prescod P. Some thoughts about SOAP versus REST on Security,
http://www.prescod.net/rest/security.html
[22] Schneier B., Shostack A. Results, Not Resolutions, 2002,
http://www.securityfocus.com/news/315
[23] SOAP Message Security 1.1, 2006,
http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
[24] Loughran S., Smith E. Rethinking the Java SOAP Stack, 2005,
http://www.hpl.hp.com/techreports/2005/HPL-2005-83.pdf
[25] Fielding R., Architectural Styles and the Design of Network-based Software Architectures, 2000
[26] Hypermedia, 1965,
http://en.wikipedia.org/wiki/Hypermedia
[27] Middleware,
http://www.sei.cmu.edu/str/descriptions/middleware.html
[28] Common Object Request BrokerArchitecture: Core Specification, 2004,
http://www.omg.org/docs/formal/04-03-01.pdf
[29] Henning M. The Rise and Fall of CORBA, 2006,
http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=396
[30] COM: Component Object Model Technologies,
http://www.microsoft.com/com/default.mspx
[31] Conformance Claim Attachment Mechanisms Version 1.0, 2004,
http://www.ws-i.org/Profiles/ConformanceClaims-1.0.html
[32] Basic Security Profile Version 1.0, 2007,
http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

[33] Web Services Addressing (WS-Addressing), 2004,
http://www.w3.org/Submission/ws-addressing/

[34] Doug Davis The hidden impact of WS-Addressing on SOAP, 2004,
http://www-128.ibm.com/developerworks/webservices/library/ws-address.html

[35] Web Services Standards as of Q1 2007, 2007,
http://www.innoq.com/resources/ws-standards-poster/

[36] Business process,
http://www.forrester.com/Research/Document/Excerpt/0,7211,39219,00.html

[37] Business Process Execution Language for Web Services Version 1.1, 2003,
http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf

[38] WS-BPEL Extension for People - BPEL4People, 2005,
ftp://www6.software.ibm.com/software/developer/library/ws-bpel4people.pdf

[39] British American Tobacco builds bridge from SOA to SAP, 2006,
http://searchwebservices.techtarget.com/originalContent/
0,289142,sid26_gci1161562,00.html?bucket=NEWS&topic=299052

[40] Stav malého a stredného podnikania 2005, 2006

[41] Zpráva o vývoji MSP a jeho podpore v roce 2005, 2006,
http://www.mpo.cz/dokument20309.html

[42] Skidata,
http://www.skidata.com

[43] Web Services Reliable Messaging TC, 2004,
http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf