VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

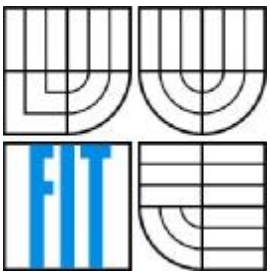
ROZHRANÍ WEBOVÝCH SLUŽEB

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE DAVID KUBÁT
AUTHOR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZHRANÍ WEBOVÝCH SLUŽEB

WEB SERVICES INTERFACE

BAKALÁŘSKÁ PRÁCE

BACHELOR´S THESIS

AUTOR PRÁCE — DAVID KUBÁT
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR WEISS
SUPERVISOR

BRNO 2007

Abstrakt

Předmětem této bakalářské práce bylo seznámit se s problematikou architektury orientované na služby a její reálnou reprezentací – webovými službami. Pochopit principy jejich činnosti, vzniku, komunikace a způsoby, jakými lze služby vyhledávat a používat.
Konkrétním cílem projektu byla analýza stávajícího programu, návrh jeho převodu do jazyka pro popis webových služeb a implementace vrstvy rozhraní webové služby nad tímto programem. Systém demonstruje základní principy webových služeb a činnost rozhraní při komunikaci mezi jejich poskytovatelem a spotřebitelem.

Klíčová slova

Webová služba, WS, rozhraní, WSI, architektura orientovaná na služby, SOA, XML, popis webové služby, WSDL, komunikační protokol, SOAP, registr služeb, UDDI, WSIL.

Abstract

The objective of this thesis was to get acquainted with the proposition of the service-oriented architecture and its real representation – web services. To understand the principles of their functioning, creation, communication and the ways to finding and using them.
To be specific, the goal of this project was to analyse a current program, design its transfer to the web service description language and to implement a web service interface layer upon this program. The system is demonstrating the basic principles of the web services and the function of the interface during the communication between its provider and customer.

Keywords

Web service, WS, interface, WSI, service-oriented architecture, SOA, XML, web service description, WSDL, communications protocol, SOAP, service registry, UDDI, WSIL.

Citace

David Kubát: Rozhraní webových služeb, bakalářská práce, Brno, FIT VUT v Brně, 2007

Rozhraní webových služeb

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Weisse
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
David Kubát
14.5.2007

Poděkování

Zde bych rád poděkoval vedoucímu práce Ing. Petru Weissovi za poskytnuté podkladové materiály a odbornou pomoc během tvorby mé práce.

Obsah

1 Úvod

World Wide Web, jak jej většina uživatelů zná, je tvořen zdroji (texty a grafikou.), určenými pro lidskou potřebu (zprávy, nabídka zboží, apod.). Programy umí tento obsah zobrazit člověku (prohlížeče), ale nedokážou v něm nalézt informaci, se kterou by mohly dále samy pracovat. Technologie pro web vyvinuté však mohou být použity i pro zpřístupnění zdrojů, které jsou strojově zpracovatelné. Bylo třeba najít způsob, jakým by mohly aplikace vzájemně spolupracovat i na velké vzdálenosti a automaticky volat své instrukce, aniž by musel do jejich konání zasahovat člověk.

Základem této myšlenky se stal značkovací jazyk XML. Pomocí něj lze vytvářet dokumenty s jasně danou strukturou, zpracovatelné nezávisle na použitém programovacím jazyce, operačním systému, či stroji. Webovým zdrojům, schopným XML dokumenty zpracovat i produkovat, se začalo říkat webové služby (Web Services).

Cílem této práce je využití znalostí získaných studiem problematiky webových služeb k vytvoření rozhraní na straně poskytovatele služeb, umožňujícího komunikaci s jeho klienty. Za tímto účelem je nutné nejdříve navrhnout jednoduchý program provádějící výpočty. Na základě tohoto programu potom vytvořit popis a návrh rozhraní webové služby nad ním postavené a na závěr vytvořit prostředí klienta pro demonstraci vzájemné komunikace a fungování celého systému.

Úvodní kapitoly této práce se zabývají teoretickým úvodem do problematiky, vysvětlením všech použitých technologií a jejich stručným popisem. Kapitoly 2 a 3 pojednávají obecně o teorii architektury orientované na služby (kap. 2) a její reálné implementaci – webových službách (kap. 3). Poté jsou krátce vysvětleny principy dvou známých a rozšířených technologií – HTTP (kap. 4) a XML (kap. 5) – na nichž jsou webové služby vystavěny. Větší pozornost je věnována technologiím, které již přímo souvisí s webovými službami. Tou asi nejdůležitější je jazyk pro popis webových služeb (WSDL) popsaný v kapitole 6 a dále potom komunikační protokol SOAP (kap. 7). Jako poslední je v kapitole 8 zmíněn registr služeb UDDI a jazyk WSIL.

Následující část (kap. 9) obsahuje podrobnou charakteristiku vlastního návrhu a zvoleného řešení. Postupně popisuje jednotlivé kroky tak, jak jimi bylo třeba během práce projít, vysvětluje proč byla zvolena která řešení a v čem spočívá jejich výhoda. Je zde také uvedeno, kde vznikaly problémy, co nebylo možné z nějakých důvodů realizovat vůbec a případně navržen možný způsob řešení. Kapitola 10 potom popisuje vlastní implementaci navrženého systému.

Závěrem práce (kap. 11) je shrnut celý její průběh, její přínos a dosažené výsledky. Jsou zde také uvedeny možnosti případných rozšíření a popsáno, kde by se dalo v práci pokračovat.

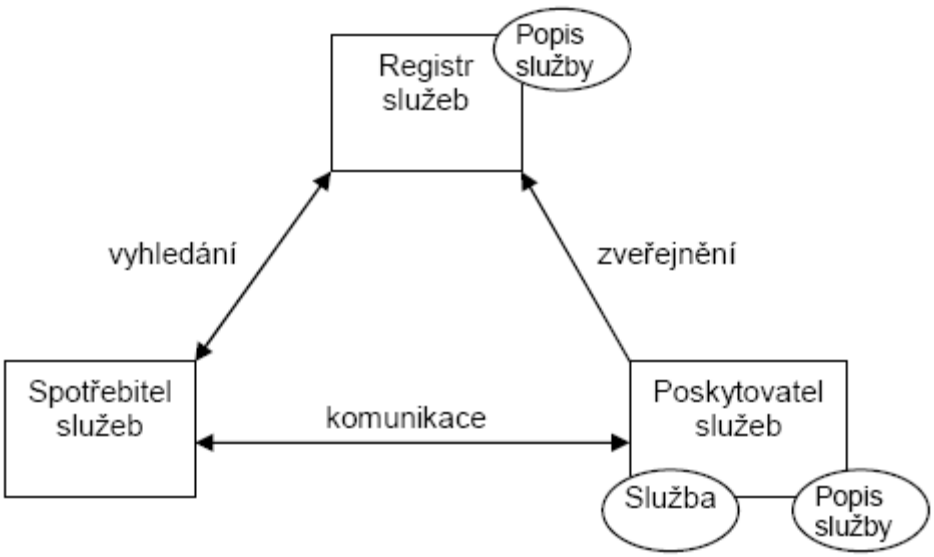
2 Architektura orientovaná na služby

Architektura orientovaná na služby (Service-Oriented Architecture, SOA) je v dnešní době jedním z nejdiskutovanějších témat na poli distribuovaných informačních systémů.

Zatím bohužel neexistuje žádná definice, která by přesně vystihovala podstatu SOA, lze však říci, že SOA je určitým stylem pro vytváření distribuovaných informačních systémů a zároveň nástrojem pro realizaci business procesů.

2.1 Konceptuální model

Nejlépe lze podstatu SOA vysvětlit na jejím konceptuálním modelu. Ten je založen na vzájemné interakci mezi dvěma klíčovými entitami: poskytovatelem a spotřebitelem služeb. Poskytovatel implementuje a nabízí služby – každá služba je specifikována svým popisem. Spotřebitel si na základě tohoto popisu vyhledá požadovanou službu v registru služeb a naváže s ní komunikaci. Model takové interakce ukazuje obrázek 2.1.



obrázek 2.1: Konceptuální model SOA

2.2 Služba

Služba se dá chápat jako mechanismus, který za běhu sestavuje své komponenty a předává jim požadavky, které jsou na ni kladeny. Pracuje na principu černé skříňky - má své rozhraní a jedině přes

něj jsou dostupné veškeré její funkce. Na základě tohoto rozhraní a dalších řídících informací vzniká popis služby, který se uplatňuje při vyhledávání a komunikaci mezi službami.

2.3 Spolupráce mezi službami

Je zřejmé, že služby jsou základními jednotkami SOA, které poskytují své prostředky buď přímo cílovému spotřebiteli anebo jiným službám. Služby mezi sebou komunikují pomocí zasílání zpráv. Spolupráci mezi službami lze rozdělit do tří kategorií: kooperace (jedna služba využívá prostředky jiné služby, aby mohla plně realizovat funkce, které nabízí), agregace (ze dvou nebo více služeb se sestaví služba nová) a choreografie (spolupráce napříč organizacemi – provedení business procesu).

2.4 SOA proti architektuře klient-server

Konceptuální model naznačuje, že SOA je určitou variantou architektury klient-server, ale ve skutečnosti je v nich velký rozdíl.

2.4.1 Architektura klient-server

Implementace této architektury se skládá ze serveru a několika klientů, kteří se k serveru připojují. Servery ukrývají logiku související s daty, zatímco klienti obsahují logiku aplikační (dvouvrstvá architektura). Veškerá data jsou tedy uložena na straně serveru, z čehož plyne větší zatížení jak serveru, tak i komunikace mezi serverem a klienty při zpracovávání požadavků.

2.4.2 SOA

U architektury SOA je situace odlišná. Každá entita, která je schopna komunikovat podle předepsaných pravidel (protokolů) je považována za spotřebitele služeb a každý spotřebitel služeb je obecně také považován za službu. Každá služba poskytuje omezenou množinu funkcí (silná distribuce), s čímž souvisí menší nároky na zdroje. Navíc je v SOA do zpráv umístěna část výpočetní logiky, což vede k nezávislé a bezestavové povaze služeb.

Rozdílem mezi oběma architekturami je také rozložení zátěže. V systému založeném na SOA je většinou více služeb poskytujících stejné funkce. Výběr služby potom uvažuje i jejich momentální vytížení.

2.5 Základní vlastnosti a výhody SOA

Nezávislost – Služby jsou autonomní sebeřídící jednotky.

Abstrakce – Služby zapouzdřují svoji logiku a okolnímu světu jsou přístupné pouze přes rozhraní.

Znovupoužitelnost – Služby mohou být navzájem propojovány tzv. ad-hoc.

Bezestavovost – Služby se snaží během komunikačních cyklů minimalizovat množství uchovávaných informací o jiných službách.

Nezávislost na platformě – Služby jsou nezávislé na implementačním jazyce i na OS.

Výhodou systémů založených na SOA je jejich způsob vývoje. Tyto systémy totiž většinou vznikají vhodnou dekompozicí stávajících systémů a využitím prostředků, do kterých bylo již dříve investováno, čímž se na vývoji nového systému ušetří.

Systémy jsou snadno rozšiřitelné a vývoj nových služeb je mnohem rychlejší a levnější, než složité úpravy stávajícího systému.

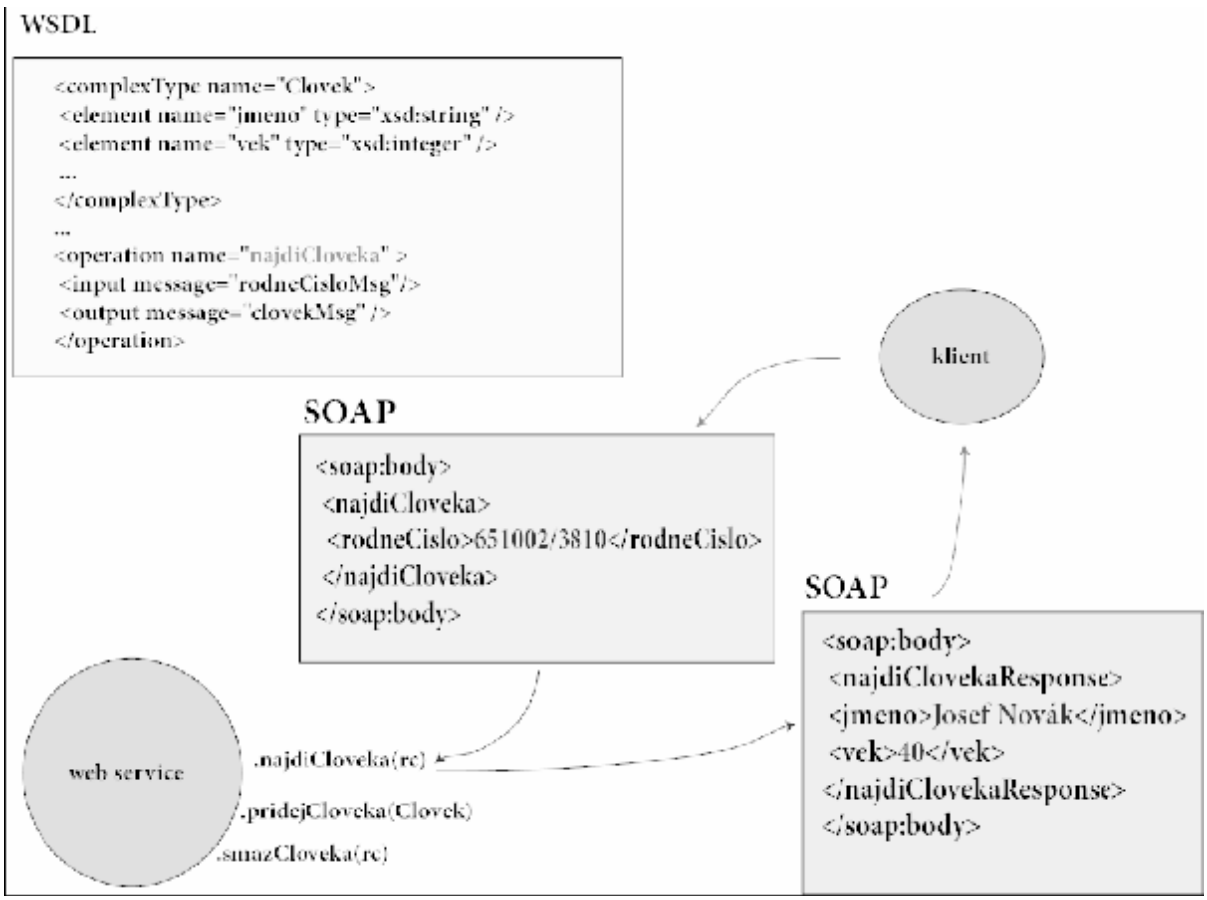
3 Webové služby

Webové služby (Web Services, WS) jsou nejznámější a nejpoužívanější implementací architektury orientované na služby. Již ze samotného názvu je patrné, že koncept ke své činnosti používá internet [1].

3.1 Definice WS

Podle dokumentu W3C, sepsaného Pracovní skupinou pro Architekturu webových služeb, zní definice webové služby takto [2]:

„Webová služba je softwarový systém zkonstruovaný k podpoře interakce mezi stroji přes síť. Má rozhraní popsané ve strojově zpracovatelném formátu (specificky WSDL). Ostatní systémy interagují s webovou službou způsobem předepsaným jejím popisem za pomoci SOAP zpráv, typicky dopravovaných použitím HTTP s XML zápisem v součinnosti s ostatními webovými standardy.“



obrázek 3.1: Princip činnosti WS

3.2 Vlastnosti WS

Nezávislost a Samostatnost – Na klientské straně není třeba žádný speciální SW, stačí pouze podpora HTTP a XML. Na straně serveru je nutný webový server a engine podporující speciální programy běžící na straně serveru (něco jako applety na klientské straně).

Samopopisnost – Definice a popis zprávy je posílán společně se samotnou zprávou.

Jelikož jsou WS podmnožinou SOA, platí pro ně i vlastnosti z kapitoly 2.5.

3.3 Technologie využívané ve WS

Webové služby k přenosu a transformaci dat využívají několik technologií založených na XML.

- **XML** (Extensible Markup Language) – základ, na kterém jsou webové služby postaveny. Poskytuje jazyk pro definici dat a popis jak s těmito daty nakládat. XML reprezentuje skupinu provázaných specifikací vydaných a spravovaných organizací W3C.
- **WSDL** (Web Services Description Language) – definuje rozhraní webových služeb, datové typy, typy zpráv a vzory protokolů.
- **SOAP** (Simple Object Access Protocol) – definuje obálku webových služeb. Je to Protokol pro přenos XML dokumentů přes síť.
- **UDDI** (Universal Description, Discovery, and Integration) – registr webových služeb a mechanismus vyhledávání používaný pro třídění informací a získávání odkazů na WS.

4 Protokol HTTP

HTTP (Hypertext Transfer Protocol) je metodou používanou k přenosu nebo zprostředkování informací na internetu. Je to protokol typu dotaz-odpověď mezi klientem a serverem. Vždy dojde k výměně této dvojice zpráv a v případě pokračování komunikace se bude jednat o další, nezávislý dotaz a odpověď. Požadavek (obrázek 4.1) i odpověď (obrázek 4.2) jsou složeny z textových hlaviček, prázdného řádku a binárních dat. Typ dat je určen pomocí MIME (Multipurpose Internet Mail Extension) typů v hlavičce [5].

```
GET /wiki/Wikipedie HTTP/1.1
Host: cs.wikipedia.org
User-Agent: Mozilla/5.0
Gecko/20040803 Firefox/0.9.3
Accept-Charset: UTF-8,*
```

obrázek 4.1: HTTP dotaz

```
HTTP/1.0 200 OK
Date: Fri, 15 Oct 2004 08:20:25 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.8
X-Powered-By: PHP/4.3.8
Vary: Accept-Encoding,Cookie
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: cs
Content-Type: text/html; charset=utf-8

<dokument/>
```

obrázek 4.2: HTTP odpověď

Za hlavičkou odpovědi následuje jeden prázdný řádek (označuje její konec) a potom požadovaný dokument.

5 Jazyk XML

Jazyk XML (Extensible Markup Language) je základ, na kterém jsou webové služby postaveny. Nepoužívá se pouze jako formát zpráv, ale hlavně také jako způsob, jakým jsou služby definovány a implementovány.

XML je značkovací jazyk podobný jazyku HTML (Hypertext Markup Language) – obsahuje elementy, atributy a hodnoty, ale na rozdíl od HTML jich umožňuje definovat neomezený počet. Je to stromová struktura s právě jedním kořenem, kde uzly stromu tvoří tagy, listy mohou být tagy, atributy i hodnoty a lze jej vyjádřit jako obecně orientovaný graf.

Tato flexibilita s sebou přináší i řadu úskalí. Vzhledem k tomu, že XML umožňuje definovat vlastní elementy, je velmi obtížné zajistit, aby byly chápány všemi stranami stejným způsobem. Řešení se nachází v XML schématech. Schémata obsahují definice a význam jednotlivých elementů a pokud obě komunikující strany sdílí kromě XML dokumentů také příslušné schéma, mohou si být jisty, že také budou elementy používat a rozumět jim stejným způsobem. Přesně toto je princip jakým webové služby fungují [1, 5].

```
<?xml version="1.0" encoding="UTF-8"?>
<recept jméno="chleba" čas_přípravy="5 minut" čas_vaření="3
hodiny">
  <titulek>Jednoduchý chleba</titulek>
  <přísada množství="3" jednotka="šálky">Mouka</přísada>
  <přísada množství="0,25" jednotka="unce">Kvasnice</přísada>
  <přísada množství="1,5" jednotka="šálku">Horká voda</přísada>
  <přísada množství="1" jednotka="kávová lžička">Sůl</přísada>
  <instrukce>
    <krok>Smíchejte všechny přísady dohromady a dobře
prohněťte.</krok>
    <krok>Zakryjte tkaninou a nechejte hodinu v teplé
místnosti.</krok>
    <krok>Znovu prohněťte, umístěte na plech a pečte v
troubě.</krok>
  </instrukce>
</recept>
```

obrázek 5.1: Příklad XML dokumentu

5.1 Technologie využívané v XML

Jazyk XML je vlastně celou rodinou technologií (značkovací jazyk, různé modely obsahu, jmenný prostor apod.). Následující pojmy jsou asi nejdůležitějšími články, které tvoří základ webových služeb:

- **XML v1.0** – Pravidla pro definici elementů, atributů a tagů obsažených uvnitř kořenového elementu, poskytující abstraktní model dat.
- **XML schema** – dokument XML definující datové typy, obsah, struktury a povolené elementy v příslušném XML dokumentu.
- **XML namespaces** – unikátní jména elementů v XML dokumentech a aplikacích
- **XSLT** (Extensible Stylesheet Language Transformations) – Převod mezi XML dokumenty nebo převod do jiných formátů.
- **DOM** (Document Object Model) a **SAX** (Simple API for XML) – programové knihovny a modely pro práci s XML dokumenty.

5.2 Vlastnosti XML

Standardní formát pro výměnu dat – data mohou být posílána a zpracovávána nezávisle na aplikaci a operačním systému.

Otevřený standard – specifikace je volně k dispozici (W3C).

Mezinárodní podpora – používá se 32b. sada.

Flexibilita a Rozšiřitelnost – pro přidání nové informace do dokumentu stačí vytvořit novou značku.

5.3 Zpracování XML dokumentů

Pro usnadnění práce s XML dokumenty je možné použít pomocného aplikačního rozhraní. Dvěma nejznámějšími jsou DOM (Document Object Model) a SAX (Simple API for XML).

Rozdíl mezi oběma je ten, že SAX prochází dokument pouze jedenkrát (tedy se hodí například pro načtení nastavení aplikace z XML dokumentu) a DOM umožňuje procházet dokument po částech a opakovaně (vhodné pokud používáme XML dokument například ve smyslu databáze). SAX je tedy rychlejší a méně náročný na paměť.

Obě implementace jsou volně dostupné pro Microsoft i Java programovací prostředí a dohromady poskytují velmi efektivní nástroj pro zpracování XML dokumentů.

6 Jazyk WSDL

Jazyk WSDL (Web Service Description Language) je jazykem určeným pro popis rozhraní webových služeb. Tento popis se ukládá do XML dokumentů (obrázek 6.1) a popisuje rozhraní na syntaktické úrovni (stejně jako .h soubory v jazyce C nebo interface v jazyce Java). Jazyk je standardizován organizací W3C, v současné době ve verzi 2.0 [1, 2, 3].

WSDL slouží k zodpovězení následujících otázek:

- Jaké funkce daná služba poskytuje? – jména dostupných operací (funkcí), jména a typy jejich parametrů a návratových hodnot.
- Kde je daná služba uložena (dostupnost)? – HTTP / HTTPS / SMTP, port, stroj, URL
- Jak může být s danou službou navázána komunikace?

6.1 Abstraktní popis

Tento popis udržuje integritu popisu služby. Obsahuje informace charakterizující její rozhraní bez ohledu na technologie, kterými bude implementována, operačním systému, ve kterém bude pracovat a způsobu komunikace.

Obsahuje čtyři základní oddíly: types, message, operation a port type.

6.2 Konkrétní popis

Konkrétní popis slouží k navázání logiky popsané abstraktním modelem na reálnou implementaci a k navázání komunikace přes konkrétní protokol.

Tento popis obsahuje oddíly: binding, service a port.

6.3 Popis jednotlivých oddílů

- tag **<types>** - popis datových typů ve formě XML schémat nebo pomocí jiných mechanismů.
- tag **<message>** - komunikační zpráva volání a návratu z operací.
- tag **<operation>** - abstraktní popis operace (jméno, input, output,…).
- tag **<portType>** - abstraktní popis celé sady operací
- tag **<binding>** - definuje možnosti přístupu různými protokoly
- tagy **<service>** a **<port>** - definují příslušnou adresu pro spojení každého protokolu

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name=" "
 targetNamespace=" "
 xmlns:tns=" " . . . >

<!-- definice typů -->
<types>
 <schema targetNamespace=" " . . . >
  <element name=" " type=" "/>
  <element name=" " type=" "/>
 </schema>
</types>

<!-- komunikační zprávy -->
<message name=" Request">
 <part name=" " element=" "/>
</message>
<message name=" Response">
 <part name=" " element=" "/>
</message>

<!-- dostupné operace -->
<portType name=" ">
 <operation name=" ">
  <documentation> </documentation>
  <input message=" Request"/>
  <output message=" Response"/>
 </operation>
</portType>

<!-- volatelné přes HTTP -->
<binding name=" " type=" ">
 <SOAP:binding style=" " transport=" "/>
 <operation name=" ">
  <SOAP:operation style=" " soapAction=""/>
  <input>
   <SOAP:body use=" " namespace=" "/>
  </input>
  <output>
   <SOAP:body use=" " namespace=" "/>
  </output>
 </operation>
</binding>

<!-- adresy komunikačních bodů -->
<service name=" ">
 <documentation> </documentation>
 <port name=" " binding=" ">
  <SOAP:address location=" "/>
 </port>
</service>
</definitions>
```

obrázek 6.1: Struktura WSDL dokumentu

7 Protokol SOAP

Jak již bylo zmíněno, veškerá komunikace mezi službami probíhá pomocí posílání zpráv. Proto musel být zaveden standard, který by umožnil službám mezi sebou komunikovat jednotným způsobem. Tímto standardem se stal protokol SOAP (Simple Object Access Protocol). Tento protokol je nezávislý na typu sítě, podporuje zprávy ve formátu XML a umožňuje spotřebiteli služeb komunikovat s jejich poskytovatelem.

Původně byl protokol navržen firmami Microsoft a IBM pro vzdálené volání procedur. V současné době je ve specifikaci 1.2 organizace W3C [1, 2].

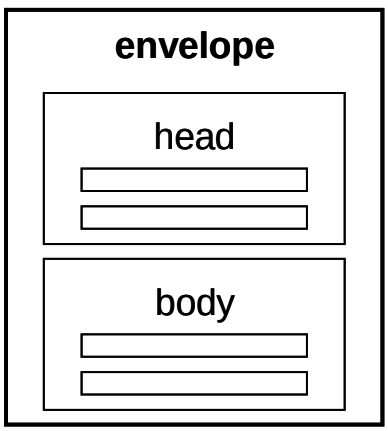
7.1 Struktura zpráv SOAP

Každá zpráva dodržující podmínky tohoto protokolu je prakticky obálkou (envelope), která zapouzdřuje hlavičku (head) a tělo (body) zprávy (obrázek 7.1).

Hlavička je nepovinná a může obsahovat několik bloků metainformací (komunikační logika, nová rozšíření). Typicky obsahuje informace nutné pro všechny služby, které mohou zprávu obdržet. Ty potom na základě těchto informací rozhodnou o způsobu, jakým bude zpráva zpracována.

Tělo obsahuje samotná data ve formátu XML. Volitelně může obsahovat sekci pro chyby (faults) a logiku pro jejich zpracování. Tělo zprávy (obrázek 7.2) obsahuje hlavní značku, pojmenovanou stejně jako procedura, která se má volat. Vnořené značky představují parametry volání. Zpráva přinášející odpověď (obrázek 7.3) v těle obsahuje opět jednu hlavní značku, jejíž název je složen z názvu volané procedury a řetězce „Response“. Vnořené značky pak obsahují návratové hodnoty.

Protokol zahrnuje i prostředky pro posílání dat, která není možné popsat pomocí jazyka XML (např. obrázky). Ty se posílají jako přílohy (attachments).



obrázek 7.1: Struktura zprávy protokolu SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <env:Header/>
 <env:Body>
  <jePrvocislo xmlns="urn:mojeURI">
    <cislo xsi:type="xsd:long">1987</cislo>
  </jePrvocislo>
 </env:Body>
</env:Envelope>
```

obrázek 7.2: SOAP dotaz

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <env:Body>
  <jePrvocisloResponse xmlns="urn:mojeURI">
   <vysledek xsi:type="xsd:boolean">true</vysledek>
  </jePrvocisloResponse>
 </env:Body>
</env:Envelope>
```

obrázek 7.3: SOAP odpověď

SOAP je vyvíjen se zaměřením na svoji jednoduchost a snadnou rozšiřitelnost. Je to bezestavový protokol nezávislý na operačním systému a transportních protokolech (HTTP, email, apod.).

8 Vyhledávání WS – UDDI a WSIL

Poté, co je webová služba hotova, je třeba nějakým způsobem klientům umožnit tuto službu najít a využívat. Pro tento problém byla navržena dvě řešení - registr služeb UDDI (Universal Description, Discovery, and Integration) a jazyk WSIL (Web Services Inspection Language) [1, 2].

8.1 Registr UDDI

UDDI není nic jiného než další webová služba, která udržuje jakýsi „telefonní seznam“ webových služeb. Je zde možné vyhledávat (obrázek 8.1) podle mnoha různých kritérií (klíčová slova, obory, apod.).

Záznamy do rejstříků mohl přidávat prakticky kdokoliv, což pravděpodobně vedlo k faktu, že většina záznamů byla špatná až nesmyslná a v lednu roku 2006 všichni tři hlavní provozovatelé UDDI rejstříků – IBM, Microsoft a SAP – své veřejně přístupné rejstříky vypnuli. Specifikace UDDI nebyla nikdy ani standardizována a v dnešní době se používá maximálně vnitropodnikově.

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <get_businessDetail generic="2.0" xmlns="urn:uddi-
org:api_v2">
      <businessKey="C90D731D-777D-4130-9DE3-5303371170C2">
    </get_businessDetail>
  </Body>
</Envelope>
```

obrázek 8.1: Příklad UDDI dotazu

8.2 Jazyk WSIL

Na neúspěchu UDDI měl patrně podíl i fakt, že služby nalezené ve veřejném rejstříku nemají nijak garantovanou svoji kvalitu. Opačný přístup volí WSIL – jazyk pro popis služeb poskytovaných nějakou firmou (institucí). Byl definován firmami IBM a Microsoft.

WSIL počítá s tím, že klient si nejdříve vybere firmu poskytující služby a z WSIL dokumentu (vždy nese název „inspection.wsil“ a je umístěn v kořenovém adresáři web serveru příslušné instituce) potom zjistí, jaké služby poskytuje.

9 Rozhraní webových služeb

Tolik tedy k teorii a nyní následuje popis řešení. Mým úkolem bylo navrhnout systém demonstrující základní principy a činnost webových služeb, s důrazem na návrh a implementaci jeho rozhraní na straně poskytovatele. Celý problém se dělí na několik celků, které spolu souvisí a vzájemně vyplývají jeden z druhého. Návrh těchto celků a problémy s ním související jsou popsány v této kapitole.

9.1 Aplikace provádějící výpočty

Nejdříve bylo třeba navrhnout program, který bude tvořit jádro, nad nímž bude rozhraní webové služby implementováno. Takovému programu se říká „komponenta“ webové služby (kap. 2). Vzhledem k demonstrační povaze celého systému se ideálním příkladem stala jednoduchá kalkulačka. Jedná se o aplikaci provádějící výpočty několika jednoduchých matematických operací. Tato aplikace je navržena tak, aby obsahovala několik vnitřních funkcí se vstupními a výstupními hodnotami.

9.2 Návrh popisu v jazyce WSDL

Dalším krokem v celém procesu bylo na základě této aplikace navrhnout popis vznikající webové služby v jazyce WSDL a tento popis formálně zapsat do dokumentu ve formátu XML. Tato část práce je asi nejobtížnější, ale zároveň se jedná o klíčovou část celého návrhu. Postupně je třeba daný program důkladně analyzovat a na základě této analýzy sestavit dokument v jazyce WSDL. Vzhledem k faktu, že jsem demonstrační aplikaci sám programoval, nebylo nutné se některými kroky analýzy nějak dopodrobna zabývat, ale pro úplnost a obecný popis celého postupu vše v následujícím textu popíši tak, aby bylo možné jej aplikovat na libovolný program.

9.2.1 Analýza I/O chování programu

Prvním krokem je tedy analýza celého programu jako celku. Je třeba zjistit co aplikace umí a nastudovat strukturu jejího zdrojového kódu. Výsledkem této činnost by měl být seznam funkcí, které aplikace nabízí.

V mém konkrétním případě to byly následující čtyři:

- Součet
- Součin
- Mocnina
- Faktoriál

9.2.2 Analýza I/O funkcí

V druhém kroku analýzy je třeba podrobně prostudovat všechny tyto funkce a z nich zjistit datové typy a počty vstupních a výstupních parametrů pro každou z nich.

Následující seznam ukazuje informace získané analýzou programu kalkulačky:

funkce pro **Součet** - 2 vstupní parametry typu `int`
- 1 výstupní parametr typu `long`

funkce pro **Součin** - 2 vstupní parametry typu `int`
- 1 výstupní parametr typu `long`

funkce **Mocnina** - 2 vstupní parametry typu `int`
- 1 výstupní parametr typu `long`

funkce **Faktoriál** - 1 vstupní parametr typu `int`
- 1 výstupní parametr typu `long`

Je také třeba vzít v úvahu jeden parametr typu `int` na vstupu celé aplikace, určující funkci, která se bude vykonávat.

9.2.3 Sestavení WSDL dokumentu

Posledním krokem při vytváření popisu v jazyce WSDL je jeho vlastní sestavení z informací a hodnot získaných během analýzy. Ač je formát tohoto dokumentu a standardizován a použité značky pevně dány, je tento proces díky své komplexnosti poměrně složitý a vyvolává řadu nejasností.

Postup a řešení uplatněné v mém projektu byly navrženy podle nastudované teorie popsané v kapitole 6. V jednotlivých blocích jsou postupně uvedeny základní definice popisu služby (definitions), definice použitých datových typů (types), definice formátu komunikační zprávy (message), souhrn operací (portType) a abstraktní popis každé z nich (operation), popis komunikačních protokolů (binding) a adres pro jejich připojení (service, port). Celý zápis odpovídajícího WSDL dokumentu se nachází v příloze 2.

9.3 Návrh komunikačního protokolu

Následující práce spočívala v návrhu komunikačního protokolu. Jak již bylo zmíněno v teoretické části (kap. 7), standardem pro komunikaci mezi webovými službami je protokol SOAP. Formát jeho zpráv vyplývá nejen z jeho specifikace (ukázky v téže kapitole), ale také právě z WSDL popisu služby.

Každá zpráva se skládá z obálky (envelope), která obsahuje definice prostředí, a z těla zprávy (body). V těle jsou v příslušných elementech obsaženy předávané informace (název operace, vstupní/výstupní hodnoty, apod.). V případě, že při výpočtu dojde k chybě (přetečení) nebo jsou

vstupní hodnoty mimo definovaný rozsah (kap. 10.1), nese tělo odpovědi namísto výsledku informaci o této chybě - konkrétně kód chyby (faultcode) a její popis (faultstring).

Vzhledem k jednoduchosti příkladu nebylo nutné zapisovat hlavičku (head) s bloky metainformací. Hlavička je ve zprávě nepovinná a tudíž tento fakt nijak neporušuje standard protokolu. Příklady příchozí, odchozí i chybové zprávy jsou uvedeny v příloze 3.

9.4 Registr služeb

Posledním krokem návrhu by bylo řešení komunikace s registrem služeb. V reálném prostředí a systému by bez jeho implementace neměla vzniknuvší webová služba ani žádný smysl, neboť by se nikde nevyskytoval její popis, nikdo by ji nemohl najít a ani využívat.

Bohužel v dnešní době jsou webové služby teprve ve stádiu svého vývoje a některé jejich technologie se osvědčily jako funkční a vhodné a jiné jako nevyhovující a chybně navržené. Systémy registrů služeb bohužel patří do druhé z těchto kategorií a tak, po vypnutí a zklamání z registrů UDDI, ani systém založený na dokumentech v jazyce WSIL nezaznamenal větší úspěch (kap. 8). Prozatím tedy neexistuje žádný standard týkající se registrování, třídění a vyhledávání služeb a tudíž ani já jsem tuto část ve svém projektu neimplementoval.

Podle myšlenky registrů UDDI by se mělo jednat pouze o další webovou službu udržující pomyslné „zlaté stránky“ služeb. Proto by komunikace s ní byla opět založena na protokolu SOAP a zprávy by měly obdobný formát jako je ten, navržený v kapitole 9.3. V samotném programu jsem tedy tuto komunikaci na příslušném místě v klientovi pouze naznačil.

10 Implementace

V této kapitole bych rád postupně popsal implementaci jednotlivých součástí systému a poukázal na některé problémy a řešení, které se během ní vyskytly.

Celý systém je, dle zadání, naprogramován v jazyce C a je určen a testován pro prostředí operačního systému MS Windows XP. Jako vývojové prostředí jsem zvolil program Dev-C++ ve verzi 4.9.9.0.

10.1 Kalkulačka

Program kalkulačky (**`kaluklacka.c`**) tvoří základ, na kterém je rozhraní webové služby implementováno. Původně vznikl jako samostatná aplikace. Po návrhu a implementaci rozhraní byl lehce pozměněn pro vzájemnou komunikaci.

Zdrojový kód kalkulačky nepřináší nic obzvláště zajímavého, ani jsem při jeho implementaci nenarazil na větší problémy. Jedná se o jednoduchý program umožňující výpočet čtyř matematických operací – součtu, součinu, mocniny a faktoriálu. Jedinou zajímavostí snad může být fakt, že výpočet faktoriálu a mocniny je realizován funkcemi, jen pomoci operace násobení namísto použití knihovní funkce jazyka C pro jejich výpočet.

Vstupní hodnoty tvoří kladná celá čísla přebírána a uložena do datového tipu **`int`**. Jejich maximální hodnota je tedy 32767. Výsledky jsou uloženy a předávány v datovém typu **`long`**. Maximální hodnotou výsledku tedy je číslo 2147483647 a vzhledem k povaze operací a vstupních čísel jsou výsledky opět vždy kladná celá čísla. Z tohoto důvodu je maximální vstupní hodnotou pro faktoriál číslo 12 a čísla pro výpočet mocniny je také třeba volit přiměřeně. Při přetečení vrací program hodnotu -1.

10.2 Strana poskytovatele

Program běžící na straně poskytovatele (**`main.c`** -> **`SOAP.exe`**) je asi nejsložitější částí celého systému, i když při jeho běhu nic k vidění není. Tento program prakticky reprezentuje implementaci celého rozhraní webové služby.

Po svém spuštění program provede inicializaci a otevře komunikační sockety (o komunikaci více v kapitole 10.4). Po této úvodní „proceduře“ program přejde do nekonečné smyčky, ve které přijímá, vyřizuje a vrací odpovědi na příchozí požadavky od klientů. Postupně jsou zde tedy volány 4 funkce pro přijetí zprávy, její zpracování, vytvoření odpovědi a její odeslání, z nichž zajímavé jsou právě prostřední dvě. Funkce **`zpracujZpravu()`**, která v poměrně složitém algoritmu z příchozí zprávy ve formátu XML vyjme požadované informace (matematickou operaci a vstupní čísla). A

funkce **vytvorOdpoved()**, která vypočtené výsledky zasazuje do XML tagů a připravuje tak zprávu odpovědi. Právě mezi těmito dvěma funkcemi program ověřuje přijaté vstupní hodnoty a od své komponenty – kalkulačky – získává vypočtený výsledek. Volání probíhá přes hlavičkový soubor **main.h**. Pokud jsou zadaná vstupní čísla mimo rozsah (kap. 10.1), nebo při výpočtu dojde k přetečení, je do zprávy odpovědi místo výsledku umístěn chybový kód a popis chyby.

Vzhledem k tomu, že program běží v nekonečné smyčce, je třeba jej systémově ukončit.

10.3 Klientské prostředí

Klientský program (**klient.c** -> **klient.exe**) je v některých ohledech velmi podobný tomu poskytovatelskému. I zde je třeba vytvářet a zpracovávat zprávy ve formátu XML a komunikovat.

Klient ihned po spuštění a ověření vstupních parametrů začne komunikovat s uživatelem. Je volána funkce vyzívající uživatele ke zvolení požadované matematické operace a zadání vstupních čísel. Další funkcí jsou potom tyto hodnoty vkládány mezi XML tagy a vytvářena zpráva k odeslání. Následuje funkce komunikující s poskytovatelem a vyřizující požadavek, a na závěr funkce zpracovávající odpověď (podobně složitý algoritmus jako na poskytovatelské straně). Poté co je výsledek zobrazen na výstupu si uživatel může vybrat, zda chce v práci pokračovat a nebo skončit.

V programu je také implementována funkce pro tisk nápovědy aktivovaná spuštěním aplikace s parametrem **-h**.

10.4 Komunikace

Komunikace mezi klientem a poskytovatelem (a naopak) probíhá pomocí tzv. schránek (socketů) na protokolu TCP/IP. Sockety jsou mechanismem pro komunikaci, který se poprvé objevil v operačním systému BSD a při „přesunu“ na operační systém MS Windows došlo k jeho mírným změnám a omezením [4].

Poskytovatel služeb (server) při svém startu vytvoří naslouchací socket s daným číslem portu (zde 6686), na kterém očekává připojení klienta. Po navázání spojení přijme od klienta zprávu a po jejím zpracování pošle zpět klientovi odpověď. Poté se naslouchací socket opět uvolní a čeká na další požadavek. Z výše popsaného textu plyne, že server nepracuje konkurentně, tedy že v jeden okamžik je schopen obsluhovat nejvýše jednoho klienta.

Klient se k serveru připojí až ve chvíli, kdy má načteny všechny vstupní hodnoty a připravenu zprávu k odeslání. I on musí připravit komunikační socket a navíc vložit IP adresu serveru (zde adresa lokálního počítače: 127.0.0.1). Po vyřízení požadavku se klient od serveru zase ihned odpojí aby jej neblokoval a až poté pokračuje ve své činnosti.

Pro správnou funkčnost komunikace pomocí socketů v operačním systému MS Windows XP je třeba k programu během kompilace nalinkovat ještě následující 2 knihovny: **libws2_32.a** a

libwsock32.a. Tyto knihovny jsou součástí vývojového prostředí Dev-C++, ale pro jistotu jsem je připojil i do adresáře se zdrojovými kódy systému. Před zahájením komunikace je navíc také třeba inicializovat prostředí WSA (funkce **WSAStartup()**).

10.5 XML tagy

Poslední součástí systému je knihovna **tagy.h** obsahující řetězce připravené k sestavování zpráv ve formátu XML. Řetězce jsou připraveny v souladu s návrhem komunikačního protokolu a popisem služby z kapitoly 9. Knihovna je společná pro poskytovatele i pro klienta a tak, v případě změny, není třeba jejich zdrojové kódy měnit, ale pouze aktualizovat tuto knihovnu.

11 Závěr

Úkolem tohoto projektu bylo prostudování problematiky webových služeb a využití získaných informací při návrhu a implementaci rozhraní webové služby. Bylo nutné podrobně nastudovat jednotlivé technologie související s touto problematikou, zejména pochopit základní principy jazyka pro popis webových služeb (WSDL) a komunikačního protokolu SOAP.

Celkový průběh práce lze rozdělit do dvou částí – teoretické a praktické. Teoretická část byla bezpochyby delší a náročnější, ale také velice zajímavá a pro mne přínosná. Sahala od studia dané problematiky, přes analýzu požadavků a daného problému, až po návrh popisu služby a výsledného řešení.

Praktickou část potom tvořila samotná implementace rozhraní webové služby. To bylo, dle zadání, implementováno v jazyce C v prostředí MS Windows XP. Pro možnost demonstrace činnosti bylo třeba vytvořit jednoduchý program, nad kterým byla vrstva rozhraní implementována a také prostředí klienta, který službu využívá. Komunikace byla realizována prostřednictvím windows socketů.

Během vývoje systému jsem postupně zjistil, že implementační jazyk nebyl zvolen zcela šťastně, neboť bylo nutné navrhovat složité algoritmy (např. zpracování XML dokumentů), pro které jsou dnes ve vyšších programovacích jazycích (Java) již běžně dostupné funkční knihovny (DOM, SAX). Dokonce i celý popis služby jazykem WSDL lze dnes generovat automatickými nástroji.

Přesto si myslím, že má práce smysl měla, prohloubila mé programovací schopnosti a dokazuje, že i problém hodný řešení ve vyšším programovacím jazyce je možné jednouše realizovat v jazyce C.

Vzhledem ke komplexnosti celé problematiky a širokému spektru jejího využití se nabízí celá řada možností jak systém rozšířit a zdokonalit. Další vývoj by měl vést především k rozšíření funkcí stávajícího systému. Hlavním krokem by bylo rozšíření množiny podporovaných matematických funkcí, popřípadě možnost zadávat složitější výrazy. K dalším rozšířením by patřila možnost zaznamenávat provoz na straně poskytovatele a vyhodnocovat jej (které funkce jsou používány častěji, které vůbec, apod.). Případně zvážit převod systému do vyššího programovacího jazyka a porovnání obou přístupů realizace řešení.

Za účelem vypracování projektu jsem nastudoval velké množství informací a seznámil jsem se s celou řadou doposud neznámých technologií. Webové služby jsou webem pro stroje a ač tak zůstávají lidským uživatelům skryty, umožňují snadný způsob komunikace mezi programy vytvořenými nejrůznějšími způsoby a patřícími různým společnostem. Osobně si myslím, že webové služby mají před sebou ještě velkou budoucnost a jsem rád, že jsem se mohl o této problematice hodně dozvědět a vyzkoušet si ji v praxi.

Literatura

[1] Newcomer, E. Understanding Web Services: XML, WSDL, SOAP and UDDI. Canada, Addison-Wesley, 2002, ISBN 0-201-75081-3.

[2] Kuba, M. WEB Services In DATAKON 2006. Letovice, PRINT, 2006, s. 93-112.

[3] Kuba, M. Tutoriál Web Services [online]. URL `http://dior.ics.muni.cz/~makub//soap/tutorial.html`

[4] Dostál R. Sokety a C++ [online]. URL `http://www.builder.cz/serial147.html`

[5] World Wide Web Consortium (W3C) [online]. URL `http://www.w3.org/`

Seznam příloh

Příloha 1. Manuál

Systém se skládá ze dvou částí: poskytovatele a klienta webové služby. Služba provádí výpočty několika jednoduchých matematických operací. Systém je navržen a testován pro práci v prostředí operačního systému MS Windows XP.

Poskytovatel webové služby

Poskytovatel služby naslouchá a přijímá požadavky od klientů. V případě obdržení požadavku z něj získá vstupní parametry a předá je své komponentě - kalkulačce. Kalkulačka provede požadovaný výpočet a vrací výsledek, popřípadě chybovou hlášku. Tato informace je poskytovatelem opět převedena do XML formátu a odeslána zpět klientovi.

Program je možné spustit z příkazové řádky. Spouští se příkazem **`SOAP`** bez jakýchkoliv dalších parametrů. Program běží v nekonečné smyčce a je třeba jej systémově ukončit např. stisknutím kláves CTRL+C.

Klientská část

Klientská část systému po spuštění interaktivně komunikuje s uživatelem. Nejdříve musí uživatel zadat číslo požadované operace (je zde i možnost program okamžitě ukončit). Poté program vyhledá vhodnou službu v registru a žádá zadání vstupních hodnot. Po jejich potvrzení klient vytvoří XML zprávu a odešle ji na server poskytovatele, odkud poté přijme odpověď. Z odpovědi program získá požadovaný výsledek a zobrazí jej uživateli. Na závěr je zde možnost pokračovat v práci zadáním dalšího výpočtu, nebo aplikaci ukončit.

Program je možné spustit z příkazové řádky. Spouští se příkazem **`klient`** bez povinných parametrů. Program je možné spustit s volitelným parametrem **`-h`** pro výpis nápovědy.

Obsah přiloženého CD je rozepsán v souboru **`obsah.txt`** v kořenovém adresáři disku.

Příloha 2. Popis služby v jazyce WSDL

Z důvodu úspory místa a zachování přehlednosti jsou uvedeny bloky pouze pro funkci sčítání. Úseky kódu pro součin, mocninu a faktorial by se na příslušných místech tvořily obdobně.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="BakalarskaPrace"
 targetNamespace="urn:localhost"
 xmlns:tns="urn:localhost"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ns1="urn:localhost"
 xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:WSDL="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">


<!-- definice typů -->
<types>
 <schema targetNamespace="urn:localhost"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="unqualified"
     attributeFormDefault="unqualified">
  <element name="cisloA" type="xsd:int"/>
  <element name="cisloB" type="xsd:int"/>
  <element name="vysledek" type="xsd:long"/>
 </schema>
</types>


<!-- komunikační zprávy -->
<message name="soucetRequest">
 <part name="cisloA" element="ns1:cisloA"/>
 <part name="cisloB" element="ns1:cisloB"/>
</message>
<message name="soucetResponse">
 <part name="vysledek" element="ns1:vysledek"/>
</message>


<!-- dostupné operace -->
<portType name="Kalkulacka">
 <operation name="soucet">
  <documentation>Funkce Soucet</documentation>
  <input message="tns:soucetRequest "/>
  <output message="tns:soucetResponse"/>
 </operation>
</portType>
```

```
<!-- volatelné přes HTTP -->
<binding name="BakalarskaPrace" type="tns:Kalkulacka">
 <SOAP:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="soucet">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
   <SOAP:body use="literal" namespace="urn:localhost"/>
  </input>
  <output>
   <SOAP:body use="literal" namespace="urn:localhost"/>
  </output>
 </operation>
</binding>


<!-- adresy komunikačních bodů -->
<service name="BakalarskaPrace">
 <documentation>Sluzba pro matematicke operace</documentation>
 <port name="BakalarskaPrace" binding="tns:BakalarskaPrace">
  <SOAP:address location="http://localhost:6686 "/>
 </port>
</service>
</definitions>
```

Příloha 3. Zprávy protokolu SOAP

Zprávy nesoucí požadavek na výpočet součinu, mocniny nebo faktorialu by měly obdobný tvar. Stejně tak odpovědi na ně.

Požadavek

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="urn:localhost">
   <SOAP-ENV:Body>
      <ns1:soucet>
         <cisloA>24</cisloA>
         <cisloB>70</cisloB>
      </ns1:soucet>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Odpověď

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="urn:localhost">
   <SOAP-ENV:Body>
      <ns1:soucetResponse>
         <vysledek>94</vysledek>
      </ns1:soucetResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Chyba

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="urn:localhost">
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>InputError</faultcode>
         <faultstring>Byla zadana chybna cisla!</faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```