VŠB – TECHNICKÁ UNIVERZITA OSTRAVA
Hornicko-geologická fakulta
Institut geoinformatiky

Řetězení webových služeb v prostředí open source GIS

Diplomová práce

Autor: **Bc. Martin Prager**
Vedoucí diplomové práce: **Ing. Jan Růžička Ph.D.**

OSTRAVA 2007

Prohlášení

- *Celou diplomovou práci, včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu.*
- *Byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. – autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo.*
- *Beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užít (§ 35 odst. 3).*
- *Souhlasím s tím, že jeden výtisk diplomové práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o diplomové práci, obsažené v abstraktu, budou zveřejněny v informačním systému VŠB-TUO.*
- *Rovněž souhlasím s tím, že kompletní text diplomové práce bude publikován v materiálech zajišťujících propagaci VŠB-TUO, vč. příloh časopisů, sborníků z konferencí, seminářů apod. Publikování textu práce bude provedeno v omezeném rozlišení, které bude vhodné pouze pro čtení a neumožní tedy případnou transformaci textu a dalších součástí práce do podoby potřebné pro jejich další elektronické zpracování.*
- *Bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.*
- *Bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).*

V Ostravě dne 25.4.2007 ***Martin Prager***

……

Adresa trvalého pobytu diplomanta

SNP 738/24

Čadca 022 01, Slovenská Republika

ANOTACE

Cílem této diplomové práce je poukázat na možnosti řetězení webových služeb v prostředí open source GIS a tímto přispět k dalšímu výzkumu a rozvoji v oblasti servisně orientované architektury.

První část této práce se věnuje teoretickému základu řetězení webových služeb, dále se zabývá stávajícími technologiemi a nástroji, vhodnými pro jeho nasazení v praxi. Druhá část zkoumá možnost integrace vybraných nástrojů implementujících technologie řetězení webových služeb do prostředí open source GIS. Poslední část popisuje aplikaci, která byla vytvořena na základě předchozího výzkumu, spolu s ukázkou jejího praktického využití.

ANOTATION OF THESIS

The target of this diploma work is to mention for possibility of web service chaining in open source GIS enviroment, and thereby contribute in research and development at service oriented architecture area.

First part of this work attends to theoretical base of web service chaining, further attends to becoming technologies and tools suitable for its practical using. Second part investigates possibility for integration of selected tools which implements technology of web service chaining to open source GIS enviroment. Last part describes application which was created on the basis of previous research, together with demonstration its practical utilization.

ANNOTATION

Das Ziel dieser Diplomarbeit ist auf die Möglichkeiten der Verkettung von Web services in der Umgebung open source GIS anzuweisen und somit zur weiteren Forschung und Entwicklung im Gebiet der service-orientierten Architektur beizutragen.

Der erste Teil dieser Arbeit widmet sich dem theoretischen Grund der Verkettung von Web services, weiterhin beschäftigt sich mit jetzigem Zustand der Technologien und Instrumente, die für seinen Einsatz in Praxis passend sind. Der zweite Teil untersucht die Möglichkeit der Integration der ausgewählten Instrumente, die die Technologien der Verkettung von Web services in die Umgebung open source GIS implementieren. Der letzte Teil beschreibt die Applikation, die auf Grund der vorigen

Forschung gemeinsam mit Demonstration von ihrer praktischen Verwendung gebildet wurde.

OBSAH

SEZNAM ZKRATEK

API	Application Programmers Interface
B2B	Business to Business
B2C	Business to Consumer
BBOX	Bounding Box
BPEL	Business Process Execution Language
CGI	Common Gateway Interface
CORBA	Common Object Request Broaker Architecture
DOM	Document Object Model
GIS	Geografický informační systém
GML	Geography Markup Language
GUI	Graphics User Interface
HGF	Hornicko-geologická fakulta
I/O	Input/Output
JUMP	Java Unified Mapping Platform
NASSL	Network Accessable Service Specification Language
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OGC CAT	OGC Catalogue Services
PDA	Personal Digital Assistant
PHP	Personal Home Page
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SAML	Security Assertions Markup Language
SAX	Simple API for XML
SCL	SOAP Contract Language
SDL	Service Description Language
SGML	Standard Generalized Markup Language
S-JTSK	Systém jednotné trigonometrické sítě katastrální
SOA	Services-oriented Architecture
SOAP	Simple Object Access Protocol
SRS	Spatial Reference System
UDDI	Universal Description, Discovery and Integration
uDig	User-friendly Desktop Internet GIS
URI	Uniform Resource Locator
UTM	Universal Transverse Mercator Geographic Coordinate System
WfMC	Workflow Management Coalition
WFS	Web Feature Service
WFS-T	WFS-Transactional
WMS	Web Map Service
WS-CDL	Web Services Choreography Description Language
WSCO	Web Services Catalog for Orchestration Open Source
WSDL	Web Services Description Language
WSFL	Web Services Flow Language
WSIL	Web Services Inspection Language
WWW	World Wide Web
XLink	XML Linking Language
XML	eXtensible Marcup Language

XPath	XML Path Language
XSL	Extensible Stylesheet Language

1. ÚVOD

V posledních letech je čím dál více kladen důraz na využívání webových služeb. Je to právě interoperabilita, která je prostředkem webových služeb a kvůli které jsou tak vyzdvihovány. Umožňují bezproblémové slučování a spolupráci zcela heterogenních systémů, platforem, aplikací a programovacích jazyků.

S rostoucím počtem dostupných služeb a nároky na efektivnost a obtížnost řešených úloh si už mnohdy nevystačíme pouze s výsledky (zdroji) jednotlivých služeb, případně s jejich statickým propojením. Jsme nuceni začít služby řetězit dynamicky. Spojovat je dle aktuálních potřeb, možností uživatele (finance, přesnost výsledků, rychlost ap.). Existují dvě úrovně řetězení služeb, známé pod názvy orchestrace a choreografie (zaměřena více globálně). Orchestrace a choreografie je spojena s některými standardy (BPEL, WS-CDL, XLANG atd.) a organizacemi (OASIS, W3C, BPMI ap.).

Servisně orientovaná architektura (SOA) přitahuje zájem všech oblastí IT průmyslu. Poháněná standardy jako XML, webové služby a SOAP, SOA rychle proniká do hlavních chodů aplikací zásadních pro plnění business operací. A právě standardy týkající se orchestrace a choreografie jsou jedny z klíčových pro rozvoj a zavedení SOA.

Momentálně jsou k dispozici různé nástroje (komerční i open source), které tyto standardy implementují. Avšak většinou se jedná o nástroje specializované jenom k tomuto účelu s podporou pouze jednoho typu služeb.

V oblasti geoinformatiky je častým výsledkem nějaké úlohy (řetězce služeb řešícího konkrétní problém) vrstva, geoprvek atd. Z tohoto důvodu by bylo výhodné spojit již funkční GIS aplikaci s nějakým nástrojem pro řetězení služeb a umožnit uživateli např. vizualizaci výsledků a jejich okamžité zpracování.

2. CÍL PRÁCE

Cíl je rozdělen do následujících bodů:

- Prostudovat problematiku řetězení webových služeb;
- Seznámit se s jazykem BPEL;
- Vyhledat vhodné nástroje pro grafický návrh řetězení služeb;
- Posoudit možnost integrace vybraných nalezených nástrojů s vybranými open source nástroji pro GIS založených na jazyce Java;
- Integrovat vybraný nástroj pro řetězení služeb do prostředí programového vybavení pro GIS nebo v případě problémů s integrací vytvořit vlastní nástroj pro grafický návrh řetězení služeb integrovaný do prostředí programového vybavení pro GIS;
- Řetězení ověřit na vhodném praktickém příkladě.

3. WEBOVÉ SLUŽBY

Od roku 2000 vystupují do popředí webové služby. Od té doby jejich využití neustále graduje a zasahuje do všech odvětví lidských činností. Zejména v oblasti geoinformatiky nalézají značně velkého uplatnění a právě servisně orientovaná architektura je její budoucností. Vhodným skládáním některých služeb můžeme docílit rychlých a přesných výsledků, které mohlo být předtím obtížné získat, nebo to bylo dokonce nemožné.

Co si má člověk představit pod termínem webová služba? V podstatě se nejedná o nic složitého. Je to aplikace (programová komponenta) identifikovaná pomocí jedinečné adresy, která poskytuje kolekci metod, ke kterým je možno přistupovat po síti s využitím standardizovaných protokolů. Tyto metody vrací na konkrétní požadavek (request) konkrétní odpověď (response). Typ odpovědi může být buďto jednoduchý datový typ (např. řetězec, číslo) a nebo složený objekt (např. GML vrstva, SVG, mapa, video sekvence) [12]. V některých případech služba nemusí nic vracet ani mít nějaké vstupní parametry. Jednoduše se jenom zavolá a něco vykoná. Dále se může jednat o službu synchronní (na požadavek obdržíme odpověď) nebo o asynchronní (někdy není možné zaslat odpověď okamžitě a to např. buď z důvodu časově náročného zpracování, aktuálního stavu služby ap.).

Webové služby jsou reinkarnací technologií pro vzdálené volání funkcí v distribuovaných systémech, jako jsou RPC, CORBA, RMI ap. Hlavní výhodou je, že umožňují interoperabilitu programů na zcela odlišných platformách (Java, MS .NET, JavaScript, C, PHP, mobilní telefony atd.).

Celá architektura webových služeb je postavená na značkovacím jazyce XML. Technologii webových služeb je možno rozdělit do několika logických vrstev.

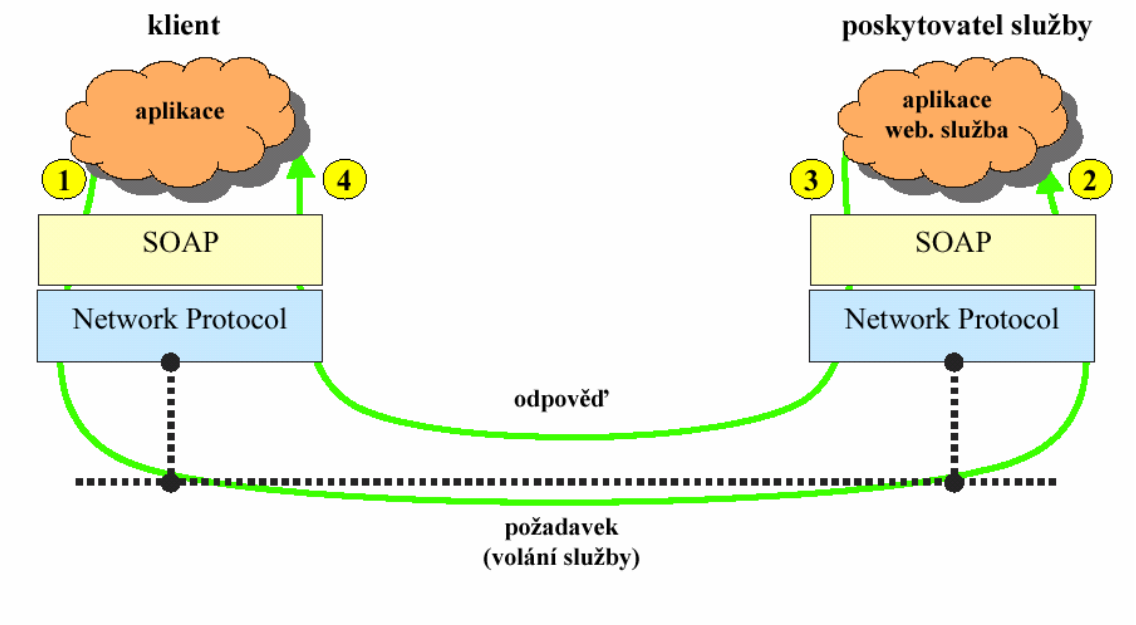
3.1. Vrstvy webových služeb

3.1.1. Komunikační vrstva

Slouží k přenosu zpráv mezi službami a klienty. Zde je nejvýznamnějším protokolem SOAP, který umožňuje již zmíněné zasílání XML zpráv mezi dvěma

aplikacemi a pracuje tedy na principu peer-to-peer. Nejčastěji se SOAP používá jako náhrada vzdáleného volání procedur (RPC), tedy v modelu požadavek/odpověď. Jedna aplikace pošle v XML zprávě požadavek druhé aplikaci, ta požadavek obslouží a výsledek zašle jako druhou zprávu zpět původnímu iniciátorovi komunikace. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy nejčastěji přes protokol HTTP přijde SOAP zpráva, spustí webovou službu a předá jí požadavek. Výsledek služby je pak předán zpět klientovi jako odpověď (viz Obrázek 1).

První verze (1.0) protokolu SOAP vznikla na konci roku 1999 jako výsledek společné práce firem DevelopMentor, Microsoft a UserLand, které chtěly vytvořit protokol pro vzdálené volání procedur (RPC), založený na XML.



Obrázek 1 - ukázka komunikace pomocí SOAP [14]

3.1.2. Vrstva popisu služby

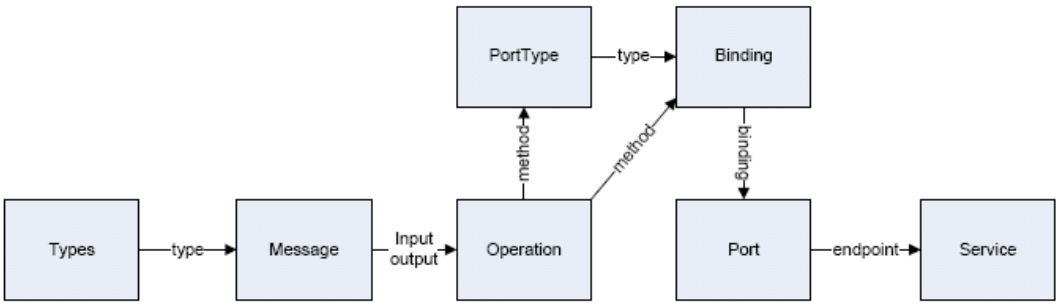
Je využívána pro popis webové služby. Dává nám odpověď na otázky, co daná služba dělá (jaké metody poskytuje, co je potřeba poslat na vstup, abychom obdrželi odpověď), jak je služba přístupná (detaily o datových typech a protokolech nezbytných pro přístup k metodám) a kde se konkrétní služba nachází (detaily o adrese specifické pro daný protokol, jako např. URL). Pro popis je nejčastěji využíván jazyk WSDL. WSDL vznikl jako společná iniciativa firem Microsoft a IBM, které si uvědomily

potřebu sjednocení jazyka, používaného pro popis rozhraní webových služeb. Navazuje tak na předchozí aktivity, zejména na jazyky NASSL, SCL a SDL.

V tomto jazyku jsou operace a zprávy popisovány na abstraktní úrovni a teprve poté jsou svázány s konkrétním síťovým protokolem a datovým formátem. To umožňuje snadné vytvoření popisu rozhraní, které nabízí jednu službu několika způsoby. WSDL soubor s definicí rozhraní služby je XML dokument. Skládá se zejména z následujících elementů, které tvoří základní části každého WSDL popisu.

- **types** – Obsahuje definici datových struktur, používaných ve zprávách. K definici lze použít teoreticky libovolný typový systém, ale nejčastěji se používají XML schémata. Nástroje pro webové služby se starají o mapování datových typů podle XML schémat na nativní datové typy použitého jazyka;
- **message** – Definuje formát předávaných zpráv pomocí dříve definovaných datových typů. Zprávy fungují jako vstupní anebo výstupní struktury pro operace. Každá zpráva se může skládat z několika logických částí s vlastním datovým typem. Při použití SOAP pro RPC odpovídá jedna část zprávy jednomu parametru vzdálené metody;
- **operation** – Abstraktní definice operací, které jsou službou podporovány. U operace se definuje jaké má vstupy a výstupy. Vstup a výstup je popsán již existující zprávou (`message`). V SOAP RPC modelu odpovídá operace metodě;
- **portType** – Sdružuje dohromady několik operací;
- **binding** – Slouží pro navázání určitého typu portu (`portType`) na konkrétní protokol a formát přenosu zpráv;
- **port** – Jeden koncový bod služby, definovaný jako kombinace síťové adresy a dříve definované vazby (`binding`);
- **service** – Sdružuje několik koncových bodů (portů) do jedné služby.

Propojení těchto elementů zachycuje následující obrázek.

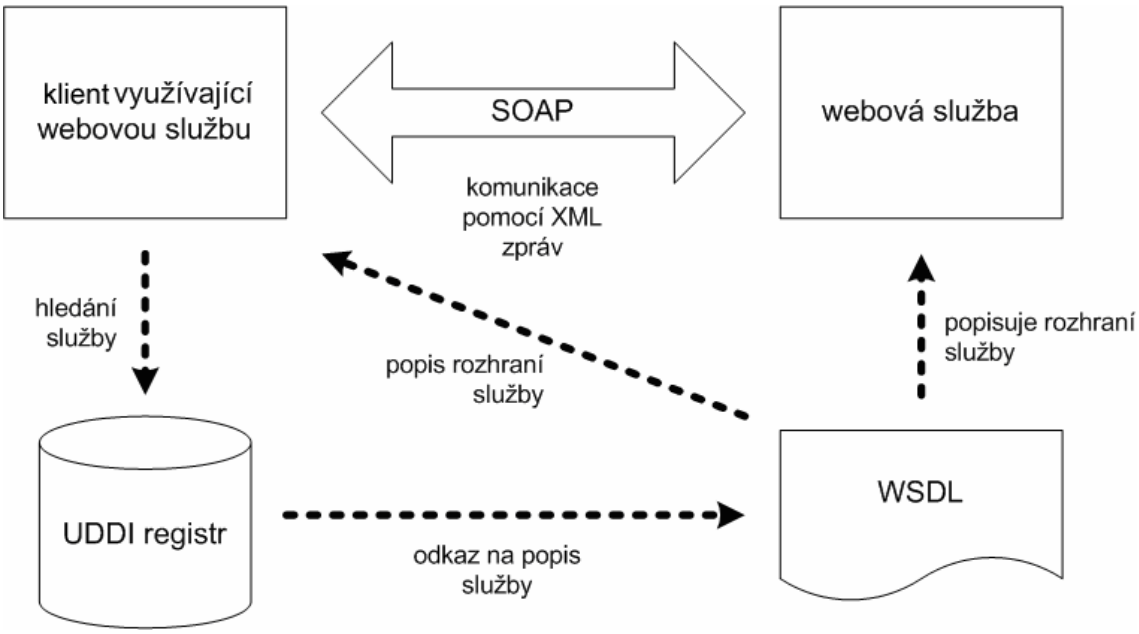


Obrázek 2 - struktura WSDL

3.1.3. Vrstva registrů

Tato vrstva poskytuje mechanizmy pro registrování, kategorizováni a hlavně vyhledávání webových služeb v reálném čase. Jednoduše řečeno, když uživatel potřebuje využít nějakou specifickou službu, prohledá daný registr. Tam získá její popis a může je začít používat. Právě zde můžeme zařadit asi jeden z nejznámějších registrů – UDDI, jedná se o standard vytvořený firmami IBM a Microsoft, který poskytuje veřejnou databázi existujících služeb. UDDI je sám o sobě službou, takže je možné v něm vyhledat další katalogy. Jelikož jde o veřejnou databázi, nelze zaručit důvěryhodnost ani relevantnost jednotlivých záznamů. Proto výše zmíněné společnosti vyvinuly další standard WSIL, který je založen na opačném principu než UDDI. V tomto případě poskytovatel nehledá klienty, ale naopak klient hledá poskytovatele a jeho služby. Popis služeb je uložen v souboru *inspection.wsil*, který se nachází vždy v hlavním adresáři web serveru poskytovatele [3]. Dále bychom zde mohli zařadit katalog OGC CAT. OGC CAT umožňuje publikovat a vyhledávat metadata pro data [10]. Bohužel tento katalog není konzistentní s ostatními katalogy, jako např. již zmíněný UDDI [13].

Obrázek 3 znázorňuje spolupráci těchto prvních tří vrstev.



Obrázek 3 - vztah tří základních technologií (SOAP, WSDL a UDDI) [7]

Právě implementací této vrstvy se zabývá projekt WSCO (viz kapitola 6.7).

3.1.4. Bezpečnostní vrstva

Bezpečnost citlivých dat je v dnešní době velmi aktuální pojem. Při posílání zpráv nezabezpečenou sítí hrozí jejich ztráta, ukradení, modifikace. Právě o toto se stará tzv. bezpečnostní vrstva, která nám přináší několik možností, jak zabezpečit komunikaci. Zde se můžeme setkat s termíny (standardy) jako WS-Security, který definuje standardní řadu rozšíření SOAP nebo rozšíření hlaviček zpráv za účelem garance budoucí výměny bezpečných a podepsaných zpráv. Dále pak XML-Signature, který zabezpečuje digitální podpisy (autentizaci) zpráv; XML-Encryption, který zlepšuje sílu XML-Signature šifrováním podepsaných zpráv. Jazyk SAML (standardizován konsorciem OASIS) je významný zejména v B2B a B2C oblasti. Slouží k výměně autentizačních a autorizačních informací (tzv. tvrzení, assertions) mezi partnery na aplikační úrovni.

3.1.5. Vrstva spolehlivosti zpráv [12]

Zajišťuje ověřování spolehlivosti přenosu zprávy a věrohodnosti, zda daná zpráva přišla ze správného zdroje a nezměněná.

3.1.6. Vrstva pro kontext, koordinaci a přenos [12]

Stará se především o koordinaci činnosti webových služeb v případě dlouhých transakcí.

3.1.7. Business Process Languages Layer

Popisuje spouštěcí logiku webových služeb definováním jejich kontrolního toku (jako podmíněné, sekvenční, paralelní, mimořádné spouštění) a předepisuje pravidla pro zpracování jejich dat.

3.1.8. Vrstva pro choreografii [12]

Umožňuje obecnější popis chování služeb při komunikaci mezi sebou (viz kapitola 4.2).

4. ŘETĚZENÍ WEBOVÝCH SLUŽEB

Poslední tři vrstvy v architektuře webových služeb jsou velmi důležité, protože se přímo týkají této kapitoly [12]. Jak již bylo zmíněno v úvodu, existují dvě úrovně řetězení služeb, známé pod názvy orchestrace a choreografie.

4.1. Orchestrace

Standardní technologie (WSDL, SOAP, UDDI atd.) pracující s webovými službami nám poskytují prostředky pro jejich jednotlivý popis, lokalizaci a spouštění. I když webová služba může poskytovat mnoho metod, každý WSDL soubor popisuje doslova atomické (na nízké úrovni) funkce. Co nám však tyto základní technologie neposkytují, jsou důležité detaily, které popisuji chování služby jako součást větší, více komplexní spolupráce. Když se jedná o spolupráci, která je kolekcí aktivit (metod, služeb) navržených tak, aby úspěšně plnila daný business cíl, jedná se o tzv. business proces. A právě popis kolekcí aktivit, který tento business proces vytváří je nazýván orchestrace [17].

Právě vytvářením již zmíněných business procesů se zabývá spousta jazyků. Například některé z následujících:

- XLANG byl představen firmou Microsoft. Jedná se o WSDL popis služby s rozšiřujícím elementem, který popisuje chování služby jako část business procesu;
- WSFL pochází od tvůrců IBM. Popisuje kompozice webových služeb a pracuje s dvěma typy těchto kompozicí. První typ (model toků), kde výsledná kompozice říká, jak dosáhnout konkrétního business cíle. Výsledkem druhého typu (globální model) je popis zahrnující všechny interakce mezi účastníky;
- Právě kombinací dvou předchozích jazyků vznikla specifikace s názvem WS-BPEL (zkráceně BPEL). Jeví se momentálně jako dominantní. Tomuto jazyku bude věnovaná vlastní kapitola.

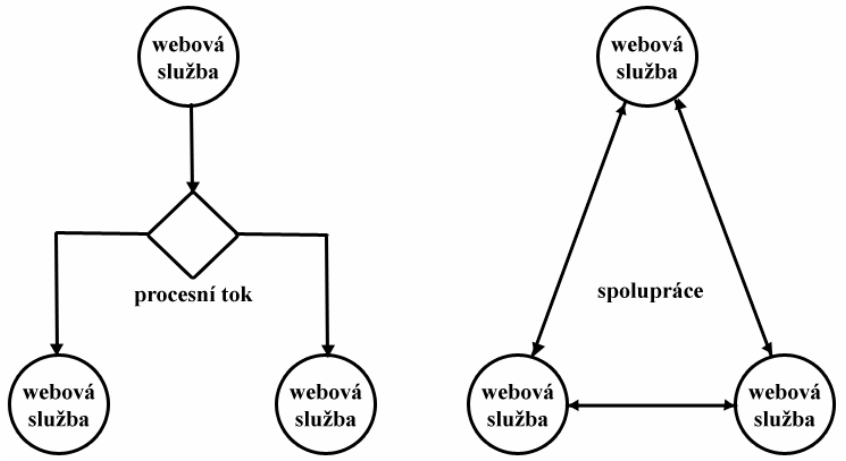
- BPML byl vyvinut neprofitující organizací BPMI (Business Process Management Initiative). Využívá BPMN (Business Process Modeling Notation) pro grafickou interpretaci procesů.

- XPDL (XML Process Definition Language) je standardizován organizací WfMC (Workflow Management Coalition). Je zajímavý tím, že obsahuje elementy sloužící přímo pro grafickou reprezentaci daného procesu.

4.2. Choreografie

Choreografie spolu s orchestrací nám poskytují otevřený, standardizovaný přístup pro spojování webových služeb. Choreografie nepopisuje žádné vnitřní akce, které se dějí uvnitř zúčastněných služeb. Zachycuje vztahy globálně (se všemi zúčastněnými službami je nakládáno rovnocenně), ve své povaze je více zaměřena na spolupráci, kde u každé zúčastněné strany v procesu je popsaná její úloha. W3C popisuje choreografii jako *„…the external observable behaviour across multiple clients (which are generally Web Services but not exclusively so) in which external observable behaviour is defined as the presence or absence of messages that are exchanged between a Web Service and it's clients*“ [19].

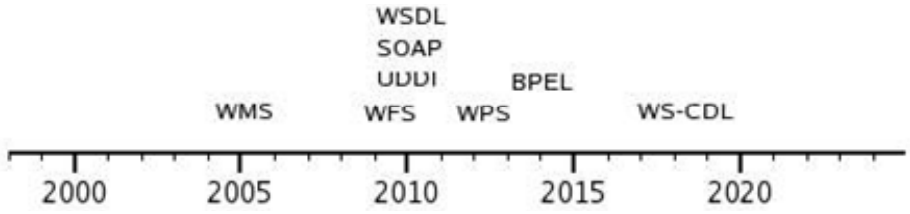
V tomto případě není pro nás tak relevantní, jaký konkrétní jazyk byl využit při orchestraci [12]. Obrázek 4 je pro lepší představu znázorňuje kontrast mezi orchestrací (vlevo) a choreografií (vpravo).



Obrázek 4 - orchestrace vs. choreografie

V této oblasti vystupuje do popředí jazyk WS-CDL (Web Services Choreography Description Language). Jedná se o W3C specifikaci, určenou pro popis peer-to-peer spolupráce účastníků z globálního pohledu.

Podle [12] není v současné době WS-CDL až tak potřebný. V budoucnosti, když bude fungovat nový business model, založený na servisně orientované architektuře, bude muset dojít ke zlepšení interoperability business procesů. Poté využití WS-CDL vypadá reálně. Kde se nachází na cestě k plně interoberabilním orchestrům momentálně geoinformatika? Odpověď poskytuje Obrázek 5.



Obrázek 5 - cesta k plně interoperabilním orchestrům [12]

5. JAZYK BPEL

Vzhledem k rozsáhlosti tohoto jazyka, je zde nastíněn pouze ve stručnosti. Tuto specifikace podrobně popisuje [9].

Jak již bylo zmíněno v předchozích kapitolách, BPEL se stal v posledních dvou letech významným standardem, vyzdvihujícím využití SOA z IT úrovně na business úroveň. Umožňuje organizacím automatizování jejich business procesů, prostřednictvím orchestrace služeb uvnitř i vně dané organizace [2]. Vyvinut firmami Microsoft a IBM, standardizován neprofitujícím konsorciem OASIS (Organization for the Advancement of Structured Information Standards). Momentálně je dostupný ve verzi 2.0. Tento jazyk umožňuje specifikovat jak spustitelné, tak abstraktní procesy.

- **Abstraktní proces** je zčásti specifikovaný proces, u kterého se nepředpokládá, že bude někdy spuštěn, ale musí být výslovně deklarován jako abstraktní. Umožňuje nám popsat úlohu, která může být použita pro více než jeden případ užití [9].

- Naopak **spustitelný proces** výlučně spoléhá na zdroje webových služeb a na XML data [9].

BPEL definuje model a prostředky pro popis chování procesu, založeného na spolupráci mezi daným procesem a jeho partnery. Spolupráce mezi všemi partnery je zprostředkovávaná rozhraními webových služeb a struktura spojení na této úrovni je zapouzdřená do takzvaného partnerLink. BPEL proces definuje jak všechny tyto spolupráce koordinovat, aby bylo dosaženo daného business cíle, stejně tak okolnosti a logiku potřebnou k tomuto dosažení. Samozřejmě zde nechybí mechanizmy pro práci s výjimkami a chybami [9].

Využívá několik XML specifikací: WSDL 1.1, XML Schema 1.0, XPath 1.0 a XSLT 1.0. WSDL zprávy a XML Schema poskytují datový model BPEL procesů. XPath spolu s XSLT ho rozšiřují o manipulaci s daty. Všechny externí zdroje a partneři jsou zde reprezentovány jako WSDL služby. Do budoucna poskytuje rozšiřitelnost těchto standardů o novější verze, obzvláště XPath a s ním spojené standardy využívané při XML transformacích [9].

Jak je z předcházejícího odstavce naprosto jasné, BPEL proces je XML dokument nejčastěji vygenerován různými grafickými návrhovými nástroji (editory). Tento návrh tvoří spíše business analytici než programátoři. Hotový proces je následně zveřejněn (zpřístupněn) pomocí nějakého konkrétního spouštěcího stroje (enginu) a to buď jako rozhraní webové služby, nebo může reagovat na spouštěcí podmínky nastavené uvnitř procesu.

5.1. Struktura procesu v BPEL

Základní struktura BPEL procesu:

```
<process...>
     <partnerLinks.../>
     <variables.../>
     <correlationSets.../>
     <faultHandlers.../>
     <compensationHandler.../>
     <eventHandlers.../>
     activities
</process>
```

Význam jednotlivých elementů je následující [9]:

- **partnerLinks** – zde jsou definovány služby, které proces využívá a poskytuje. Uvnitř každého partnerLink musí být vymezena aspoň jedna z dvou možných rolí. Roli samotného procesu určuje atribut myRole a naopak roli partnera atribut partnerRole;
- **variables** – tato část specifikuje proměnné, které proces využívá. BPEL umožňuje deklarovat proměnné třemi způsoby: jako typ WSDL zprávy, jako typ XML Schema (jednoduchý nebo složený) a jako XML Schema element;
- **correlationSets** – ještě před spuštěním daného procesu dochází k vytvoření jeho instance. Význam tohoto elementu spočívá v tom, že zabezpečuje doručování přicházejících zpráv odpovídajícím instancím procesů;
- **faultHandlers** – jak už název sám o sobě říká, tento element slouží ke zpracování chyb, které nastanou při běhu procesu. Chyby mohou být vyvolány explicitně (pomocí elementu <throw>), nebo implicitně (jako např.

výsledek chyby při volání nějaké partnerské služby). Naopak pro zachytávání chyb je využíván element s odpovídajícím jménem <catch>;

- **compensationHandler** – tento element poskytuje možnost zpětného zotavení z chyby v specifikované oblasti. Jinými slovy pokusit se o jakési vyčistění a navrácení se do stavu, kde může proces po chybě pokračovat.

- **eventHandlers** – slouží k zachycení událostí. V BPEL jsou dva typy: příchozí zprávy (korespondují s WSDL operacemi) a alarmy (aktivovány po uživatelem zadaném čase). V každém tomto elementu musí být obsažen aspoň jeden zmíněny typ.

5.2. „Aktivity“ v BPEL procesu

Jak lze vidět v 5.1, struktura procesu pokračuje tzv. aktivitami (elementy), které už implementují jeho samotný tok. Každý proces má jednu hlavní aktivitu. BPEL obsahuje sadu jednoduchých aktivit, které můžeme skládat a vytvářet tak aktivity složené.

Seznam jednoduchých aktivit a jejich stručný popis:

<receive>	Umožňuje procesu čekat na odpovídající zprávu. Je ukončená právě jejím příchodem.
<reply>	Zasílá zprávu jako odpověď na zprávu přijatou aktivitou <receive>.
<invoke>	Prostřednictvím této aktivity lze spouštět jednocestné nebo obousměrné (request-response) operace na daném portu poskytovaném partnerem.
<assign>	Používá se při manipulaci s proměnnými. Může obsahovat mnoho elementárních přiřazení jako např. kopírování nebo aktualizace.

<throw>	Aktivita umožňující generování chyb uvnitř procesu.
<exit>	Okamžité ukončení instance procesu, v kterém byla tato aktivita vyvolána.
<wait>	Používá se pro čekání procesu. Aktivita je ukončena buď po uplynutí určitého časového intervalu, nebo po dosažení určitého časového mezníku.
<empty>	Jedná se o tzv. „no-op“ aktivitu. Lze jí použít v případě, kdy je zapotřebí aktivita, která nic nedělá (např. při zachycení a ošetření chyb). Dále jí je možné využít jako synchronizační bod při paralelně běžících aktivitách.
<sequence>	Definuje kolekci aktivit, které jsou poté vykonávány sekvenčně v daném pořadí.
<if>	Vybírá jednu aktivitu ke spuštění z dané kolekce aktivit. Jedná se o klasický rozhodovací blok známý z ostatních prog. jazyků.
<while>	Vytváří cyklus. Opakování vnořených aktivit tak dlouho, dokud je daná podmínka pravdivá.
<repeatUntil>	Podobná aktivitě <while> s tím rozdílem, že opakování cyklu je ukončeno v momentě, když daná podmínka je splněna.
<forEach>	Cyklus se zadaným počtem opakování. Je možné cyklus ukončit i dříve s využitím elementu <completionCondition>. Po splnění podmínky je opakování ukončeno.
<pick>	Umožňuje čekat na jednu z několika možných příchozích zpráv, nebo určitý časový interval. Pokud jeden z těchto dvou případů

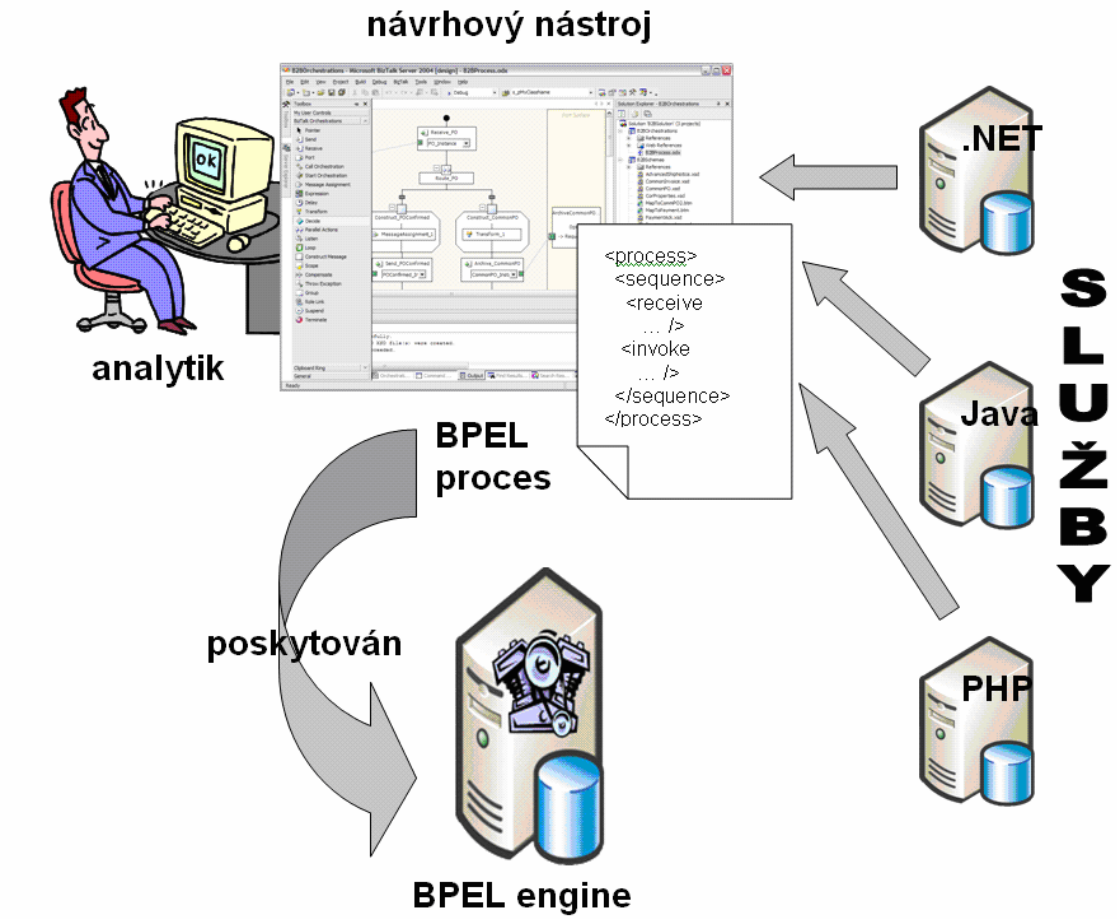
	nastane, aktivita je ukončena.
<flow>	Slouží k definování paralelních běhů aktivit. Uvnitř může být využit element <links>, který se používá pro definování jednoznačné kontroly závislostí mezi vnořenými aktivitami.
<scope>	Poskytuje jakýsi uzavřený kontext (podobný jako samotný proces). Tento kontext je sdílený všemi jeho vnořenými aktivitami. Elementy <process> a <scope> sdílejí syntaxi, která má stejnou sémantiku. Ovšem mají některé odlišnosti.
<compensate>, <compensateScope>	Slouží k vyvolání již zmíněného compensationHandler.
<rethrow>	Umožňuje opětovné vyvolání chyby, která byla původně zachycena rodičovským chybovým správcem (fault handlere). Může být použita pouze uvnitř tohoto správce.
<validate>	Prostřednictvím této aktivity lze validovat hodnoty proměnných vzhledem k jejich přidruženým WSDL a XML datovým definicím.
<extensionActivity>	Dovoluje rozšířit BPEL specifikaci o nové aktivity.

Tabulka 1 - seznam jednoduchých BPEL aktivit [9]

5.3. BPEL v praxi

Jak vypadá taková cesta BPEL procesu od jeho zrodu až po užití v praxi, znázorňuje Obrázek 6. Na začátku stojí analytik (může se jednat i o aplikaci generující na základě přesně daných pravidel zdrojový kód procesu), který daný proces navrhne. Vzhledem k tomu, že BPEL proces je obyčejný XML soubor (většinou je zapotřebí více souborů k jeho používání), může ho zmíněná osoba napsat v jednoduchém textovém editoru nebo využít některého z dostupných grafických editorů. Do procesu mohou

vstupovat různé typy webových služeb. Nakonec finální a od chyb odladěný proces je spouštěn a poskytován klientům (uživatelům) prostřednictvím tzv. BPEL stroje (BPEL engine), nejčastěji ve formě webové služby. Některé stroje mají pro toto své vlastní prostředí (klienta).



Obrázek 6 - BPEL v praxi [15]

V současnosti je k dispozici řada takovýchto komerčních i open source strojů, např.:

- **Apache Ode (Orchestration Director Engine) –** open source projekt, Java implementace, 100% podpora BPEL. Momentálně vyvíjen v inkubátoru Apache Software foundation. http://incubator.apache.org/ode/;
- **Twister –** open source stroj implementován v jazyku Java. Jeho poslední verze 0.3 byla zveřejněna v březnu 2005. Bohužel je podporován malým množstvím vývojářů a podpora BPEL je nekompletní. Funguje pod Apache Tomcat serverem [6]. http://sourceforge.net/projects/wf-twister/;

- **Apache Agila** – open source, skládá se z dvou modulů Agila BPM a Agila BPEL. Originální kód byl převzat ze zmíněného stroje Twister. Později byl tento projekt přesunut pod Apache Ode;

- **FiveSight PXE BPEL Engine –** původně stvořen jako komerční produkt, v roce 2005 přechází na open source a nakonec se také spojuje s již zmíněným Apache Ode [6];

- **IBM's BPWS4J** – komerční produkt, dostupný v 90 denní zkušební verzi. 100% podpora BPEL specifikace verze 1.1. Robustní produkt s dobrou dokumentací a vazbou přímo na vývoj této specifikace. http://www.alphaworks.ibm.com/tech/bpws4j [6];

- **ActiveBPEL engine** – opět robustní open source stroj se 100% BPEL podporou, postavený na jazyku Java, podporován komerční společností Active Endpoints, Inc., dostupný na http://www.active-endpoints.com/. Spustitelný pod Apache Tomcat a Web-Sphere Apcication serverem;

- **Bexee –** Vytvořen na Berne University of Applied Sciences jako diplomová práce. Má dobrou dokumentaci, jeho poslední verze je 0.1. Bohužel podpora jazyka je nekompletní a jeho vývoj jak se zdá už nepokračuje. http://sourceforge.net/project/showfiles.php?group_id=121714 [6];

- **Další –** Microsoft BizTalk Server, Oracle BPEL Process Manager, Sun Microsystems eInsight BPM, TIBCO Business Works, iGrafx BPEL Interface, Parasoft: BPEL Maestro, Cape Clear Orchestrator, OpenStorm ChoreoServer.

Většina strojů potřebuje mít proces spolu s popisným souborem zkomprimovaný ve své vlastní adresářové (souborové) struktuře, aby mohl být spouštěn. To znamená, že pokud bychom chtěli ten samý proces provozovat s jiným strojem, museli bychom ho překomprimovat do správné struktury a také upravit k němu příslušející popisný soubor.

Při praktické části této práce byl využíván ActiveBPEL engine (verze 3), který se zdá být jako jeden z mála open source strojů vyspělý natolik, aby se dal uplatnit pro řešení konkrétních případů (obchodních cílů) v reálném životě. Byl provozovaný pod Apache Tomcat 5.5.16.

5.4. Ukázkový proces

Jedná se o situaci, kdy uživatel chce nalézt vodní plochy v okolí dané obce do určité vzdálenosti (pouze v ČR). Do procesu vstupují dva údaje: název obce a vzdálenost v metrech. Výsledek je vrácen v podobě GML vrstvy s vodními plochami. Do procesu vstupují dvě služby a celkem využívá jejich tří metod. Tento proces by se dal popsat těmito body (které v podstatě reprezentují funkci jednotlivých metod):

- nalezení UTM souřadnic dané obce,
- transformace UTM souřadnic do formátu S-JTSK,
- návrat GML vrstvy vodních ploch na základě S-JTSK souřadnic a vzdálenosti

V příloze 14.1 se nachází zdrojový kód tohoto procesu spolu s ostatními soubory nezbytnými ke spuštění v ActiveBPEL engine. Detaily týkající se spuštění viz http://www.active-endpoints.com/documentation-deployment.htm

6. TECHNOLOGIE A NÁSTROJE PRO APLIKAČNÍ ŘEŠENÍ

6.1. Java

Java je programovací jazyk pocházející od firmy Sun Microsystems. Jedná se o objektově orientovaný jazyk vycházející z C++, ke kterému má také syntakticky nejblíže.

Nejedná se jenom o programovací jazyk ale také o platformu. Java obsahuje velmi mnoho již hotových knihoven tříd, od tříd pro vytváření grafických rozhraní, přes třídy pro přístup k databázím, až po šifrování a kompresi dat. Podobně jako u operačních systémů, např. Windows, kde existuje řada programových rozhraní pro všechny možné oblasti a přístroje, nabízí Java knihovny své funkce ve formě API. Ve skutečnosti existuje pro operační systémy jedno a to samé rozhraní a jednotlivá volání jsou za běhu převáděna do skutečné platformy. Toto abstraktně definované prostředí spolu s příslušným emulátorem se nazývá virtuální stroj (VM) [5].

Java vývojář pracující vytvoří pouze jeden univerzální zdrojový kód. Ten posléze přeloží do binárního kódu pro Java VM, tzv. mezikód (bajtkód). Spouštěcí prostředí se skládá z emulátoru a řady knihoven tříd a tento mezikód vykonává. Toto se nazývá Java Runtime Environment [5]. Tato nezávislost na platformě je její hlavní výhodou. Tedy program napsaný v jazyce Java by měl fungovat všude tam kde je k dispozici Java Runtime Environment (např.: PC s různými operačními systémy, PDA, mobilní telefony).

Java umožňuje tvorbu několika druhů programů:

- **Java applety** – jsou spouštěny jako součást WWW stránky,
- **desktop aplikace –** klasické aplikace využívající pro komunikaci s uživatelem systém oken,

- **CGI programy (skripty)** – jedná se o spustitelný program, který zpracuje předané parametry a jako výsledek na standardizovaný výstup zapíše odpověď pro WWW server,
- **Java servlety a JavaServer Pager (JSP)** – obě tyto technologie umožňují na straně serveru dynamické generování dokumentů jako HTML, XML ap. (obsah je generován jako odpověď na požadavek klienta)

Vzhledem k tomu, že Java může fungovat na různých platformách (zařízeních), nemá smysl nabízet pro každou platformu všechny knihovny. Například na mobilním telefonu určitě nebudou zapotřebí komponenty pro aplikační server. Z tohoto důvodu Sun definoval tři různé edice, které se liší vybavením:

- **Standard Edition** – pro stolové počítače,
- **Enterpise Edition** – pro aplikační servery,
- **Micro Edition** – pro malé přístroje jako mobilní telefony, mp3 přehrávače ap.

Soubor se zdrojovým kódem Javy má příponu *.java*. Název souboru musí odpovídat názvu veřejné (public) třídy, kterou soubor definuje.

6.2. XML

Extensible Markup Language je jednoduchý, velmi flexibilní textový formát, kterému dal základ jazyk SGML. Původně byl vytvořen se záměrem pro rozvoj rozsáhlého elektronického publikování. XML hraje neustále větší roli při výměně různorodých dat v prostředí webu, nebo kdekoliv jinde. 10. února 1998 se XML stává W3C specifikací [19].

XML má mnoho výhod:

- jedná se o formát čitelný jak člověkem tak strojem,
- textový formát = snadná editace,

- stejný jazyk pro popis algoritmů i dat,
- podpora Unicode,
- snadná konverze do jiných formátů

XML sám o sobě není konkrétní jazyk, je to specifikace určující, jak mají značkovací jazyky vypadat. Autor si tedy definují své vlastní značkovací jazyky. Vzhledem k tomu, že jména značek a atributu se můžou opakovat, využívá XML jmenných prostorů k jednoznačné identifikaci konkrétního jazyka.

Každý XML dokument by měl splňovat tyto pravidla:

- **musí být dobře vytvořen** (*well-formed*) – musí mít správnou syntaxi (kořenový prvek, uzavírání značek, křížení značek ap.),
- **měl by být platný** (*valid*) – dokument je validní v případě, že je asociován s určitým XML schématem, Relax NG schématem, nebo starším DTD (Document Type Definition) a přesně splňuje jeho přepis

Jak již bylo zmíněno, každý dokument se skládá ze značek a jim příslušejícím atributům.

Jednoduchý příklad:

```
<?xml version="1.0" encoding="UTF-8"?>
<osoba id="pra089" xmlns="cz.vsb.gisak.parek.diplomka">
	<jmeno>Martin</jmeno>
	<prijmeni>Prager</prijmeni>
	<pohlavi>M</pohlavi>
	<vek>25</vek>
</osoba>
```

Na prvním řádku se nachází XML deklarace, která říká, o jakou verzi se jedná a jaké kódování používá. Všeobecně instrukce v XML jsou uzavírány mezi symboly <?...?>. Pojmenování značek a atributů je na autorovi konkrétního jazyka. Atribut xmlns definuje jmenný prostor z kterého daná značka (značky, dokument) pochází.

Autor může definovat nejrůznější datové struktury a využívat spousty dalších specifikací (XSL, XPath, Xlink ap.). Možnosti jazyka jsou velmi široké (pro hlubší zájem viz [19]).

Prvním krokem ke zpracování XML dokumentu je jeho analýza, kontrola syntaxe, ověření, že dokument odpovídá danému DTD či XML schématu. Dokument je

potom převeden na interní datovou reprezentaci, kterou je možno dále zpracovat. Tuto činnost zajišťuje tzv. XML parser. XML parsery využívají pro přístup ke struktuře a obsahu dokumentu standardní rozhraní, např. SAX, DOM.

6.3. GML

Geography Markup Language je poměrně rozsáhlou specifikací konsorcia OGC. Z tohoto důvodu je zde popsán pouze ve stručnosti (v případě hlubšího zájmu viz [10]).

GML je jazyk pro modelování, přenos a ukládání prostorových dat včetně jejich prostorových a neprostorových vlastností. GML nám poskytuje pro popis geografických prvků různé typy objektů – pro souřadnicové systémy, geometrii, topologii, čas, rozměrové jednotky a generalizované hodnoty. Geografický prvek je „abstrakce úkazu (jevu) reálného světa a to v případě, že je asociován k relativní pozici na Zemi“. Takže se můžeme dívat na digitální reprezentaci světa jako na sadu těchto prvků. Stav prvku je vyjádřen sadou vlastností, kde u každé vlastnosti lze popsat minimálně jméno, typ a hodnotu [10].

Specifikace je postavena na jazyce XML a definovaná XML schématy. Jazyk je navržen se silnou objektovou orientací [10]. Momentálně je dostupný ve verzi 3, bohužel v současné době není dostatek programů podporujících tuto verzi. Verze 3.1 je i mezinárodním standardem (ISO 19136). Prozatím se ještě nejedná o finální verzi.

Jazyk je zpětně kompatibilní, avšak verze tři značně přidává na jeho složitosti a využití. Verze dvě je postavena na třech XML schématech:

- **geometry.xsd** – definuje geometrickou složku geoprvku,
- **feature.xsd** – definuje hlavní model prvek-vlastnost,
- **Xlink.xsd** – jedná se o W3C specifikaci poskytující funkce pro odkazování, zde slouží k vytváření asociací mezi objekty

GML verze tři bylo obohaceno o dalších 25 schémat.

6.4. Eclipse

Eclipse je open source komunita, jejíž projekty jsou zaměřeny na poskytování obchodně nezávislé otevřené vývojářské platformy a aplikačních rámců pro tvorbu softwaru. Tento projekt spustila firma IBM a spravuje jej Eclipse Foundation [4].

Nástroje založené na Eclipse dávají vývojářům možnost volby vícejazyčného a více-platformového prostředí. Eclipse poskytuje systém založený na zásuvných modulech, který ulehčuje vytváření, integraci a využívání programových nástrojů, šetřících čas a peníze. Tato platforma je napsána v programovacím jazyce Java a v současné době je k dispozici pro mnoho operačních systémů (Linux, HP-UX, AIX, Solaris, QNX, Mac OS X, Windows) [4].

6.5. uDig

Jedná se o desktop GIS produkt vyvíjený společností Refraction Research, napsán v jazyce Java. Je postaven na knihovně GeoTools (open source Java knihovna s důrazem na OGC specifikaci, poskytující standardní metody pro manipulaci s prostorovými daty) a na Eclipse Rich Client Platform.

Momentálně poskytuje uDig tyto možnosti [18]:

- **podpora standardních GIS formátů, tiskového výstupu, souřadnicových systémů,**
- **WFS klient** – umožňuje jak prohlížení tak editaci dat poskytovaných prostřednictvím služby WFS a WFS-T,
- **WMS klient** – prohlížení dat zprostředkovaných službami WMS
- **podpora SLD (Styled Layer Descriptor)** – rozšiřuje WMS specifikaci tím, že uživateli umožňuje vytváření barevných zvýraznění prostorových dat (určuje jaké barvy a symboly budou reprezentovat jednotlivé prvky a vrstvy)
- **podpora databází** – PostGIS, OracleSpatial, ArcSDE, MySQL atd.,

- **tvorba zásuvných modulů**

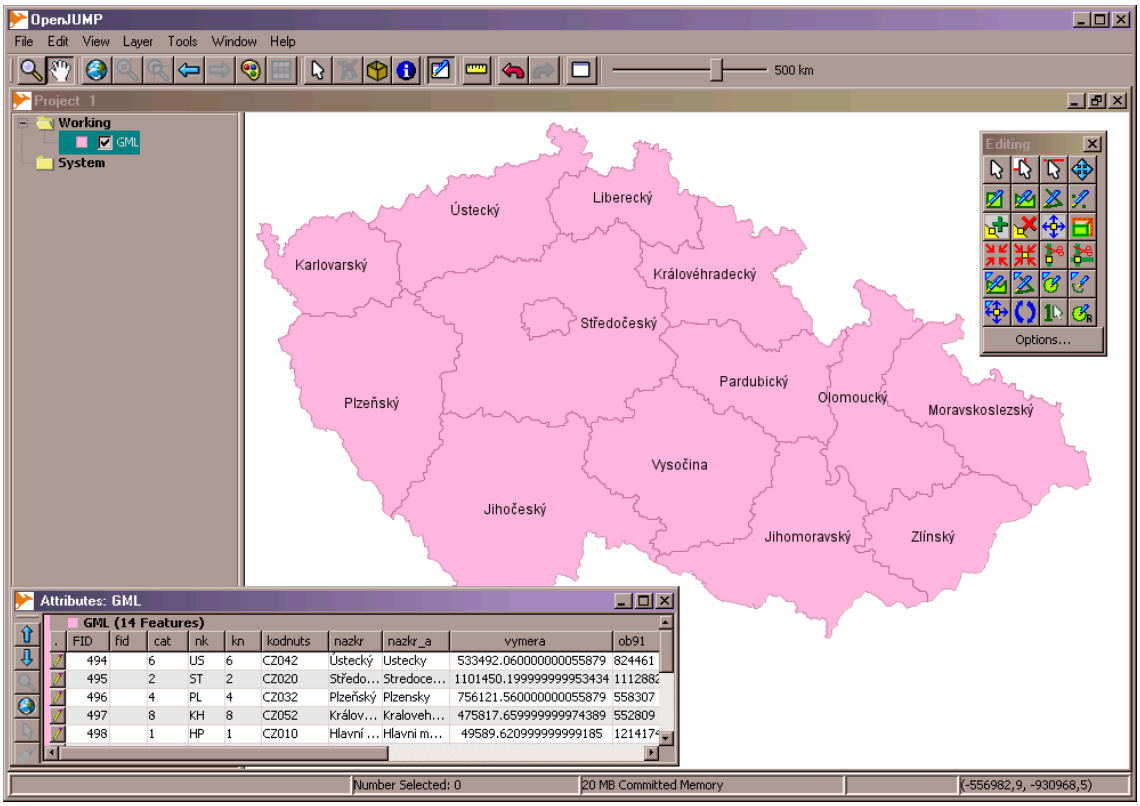
Tvorba zásuvných modulů má dobrou dokumentaci včetně vzorových příkladů (viz [18]). V rámci této práce byla prozkoumána možnost tvorby zásuvných modulů a následné její praktické vyzkoušení.

6.6. OpenJUMP

Podobně jako program uDig, také v případě OpenJUMP se jedná o open source GUI desktop GIS aplikaci, postavenou na jazyce Java. Vychází z projektu JUMP Project od společnosti Vivid Solutions Inc. (http://www.vividsolutions.com/) [11].

OpenJUMP umožňuje prohlížení, editaci a zpracování prostorových dat (podobné funkce jako u produktu uDig). Podporuje významné standardy jako např. GML, WMS. Jeho API poskytuje programátorům přístup ke všem funkcím včetně I/O rozhraní, datovým sadám, vizualizaci a prostorovým operacím. Tato možnost z něj dělá vysoce modulární a rozšiřitelný produkt [11]. Rozšíření jsou zde realizována prostřednictvím zásuvných modulů. V současné době je již k dispozici spousta takovýchto modulů, rozšiřujících jeho funkcionalitu.

OpenJUMP má velmi dobrou a názornou dokumentaci jak pro uživatelskou tak i pro vývojářskou komunitu (viz [11], nebo http://www.jump-project.org/).



Obrázek 7 - prostředí OpenJUMP

6.7. WSCO

Jedná se mnou vytvořený katalog webových služeb, který vnikl jako součást grantového projektu „*Výzkum možností metadatové katalogizace webových služeb pro účely jejich orchestrace*“. Dostupný na adrese http://gisak.vsb.cz/wsco/intranet/. Katalog je napsán s využitím jazyka PHP a ukládá data do postgreSQL databáze.

Projekt je neustále ve fázi vývoje a v současné době podporuje registraci a vyhledávání služeb typu WMS, WFS a wsdl. Registrace a vyhledávání je realizováno za pomocí vlastního klienta, nebo prostřednictvím webových služeb. Katalog by měl být v blízké budoucnosti rozšířen o standardizované UDDI rozhraní.

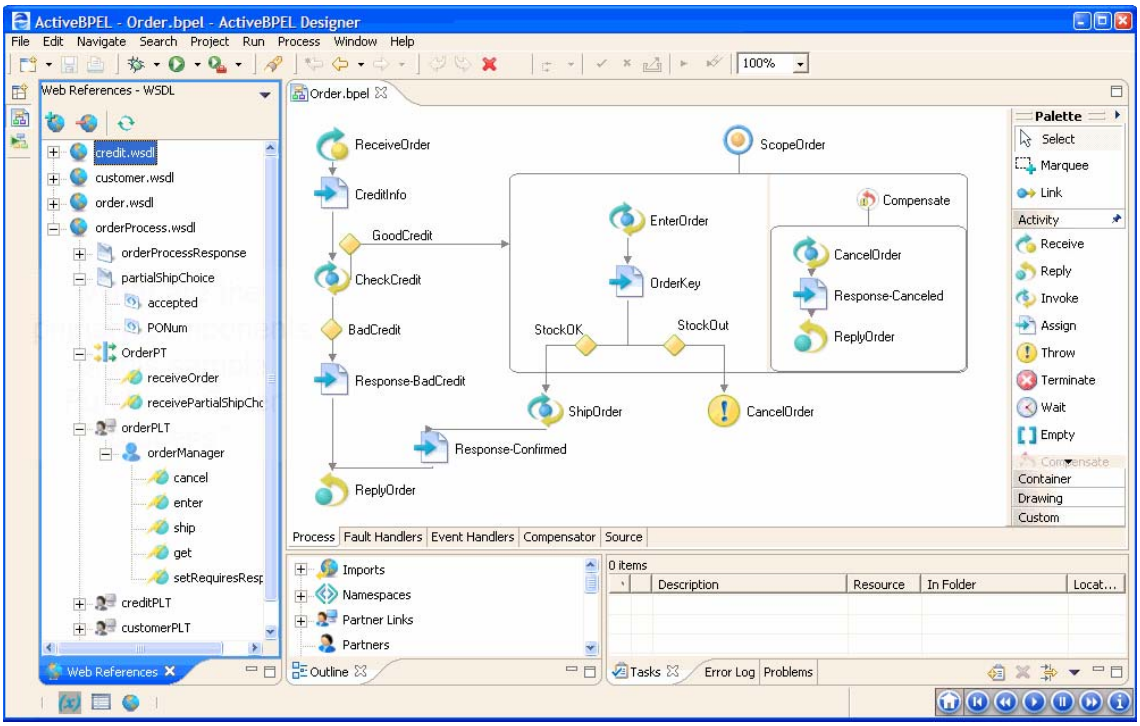
7. VHODNÉ NÁSTROJE PRO GRAFICKÝ NÁVRH ŘETĚZENÍ SLUŽEB

Při hledání těchto nástrojů hrál velkou roli fakt, aby tyto nástroje byly postavené na vývojovém prostředí Eclipse. A to z toho důvodu, že jako vhodný produkt pro implementaci se jevil program uDig.

V době (říjen 2006) hledání byly nalezeny následující editory:

7.1. ActiveBPEL Designer

Produkt pochází z dílny již zmíněné společnosti Active Endpoints, Inc. Dalo by se říct, že spolu s ActiveBPEL engine tvoří „jakýsi“ softwarový balík. Avšak paradox je v tom, že stroj je open source a designér komerční program dostupný (http://www.active-endpoints.com/product-download-form.htm) zdarma v 30 denní verzi. K získání tohoto produktu je nutná registrace a do několika dnů obdržíte email s odkazem pro stažení produktu.



Obrázek 8 - prostředí ActiveBPEL designéru [1]

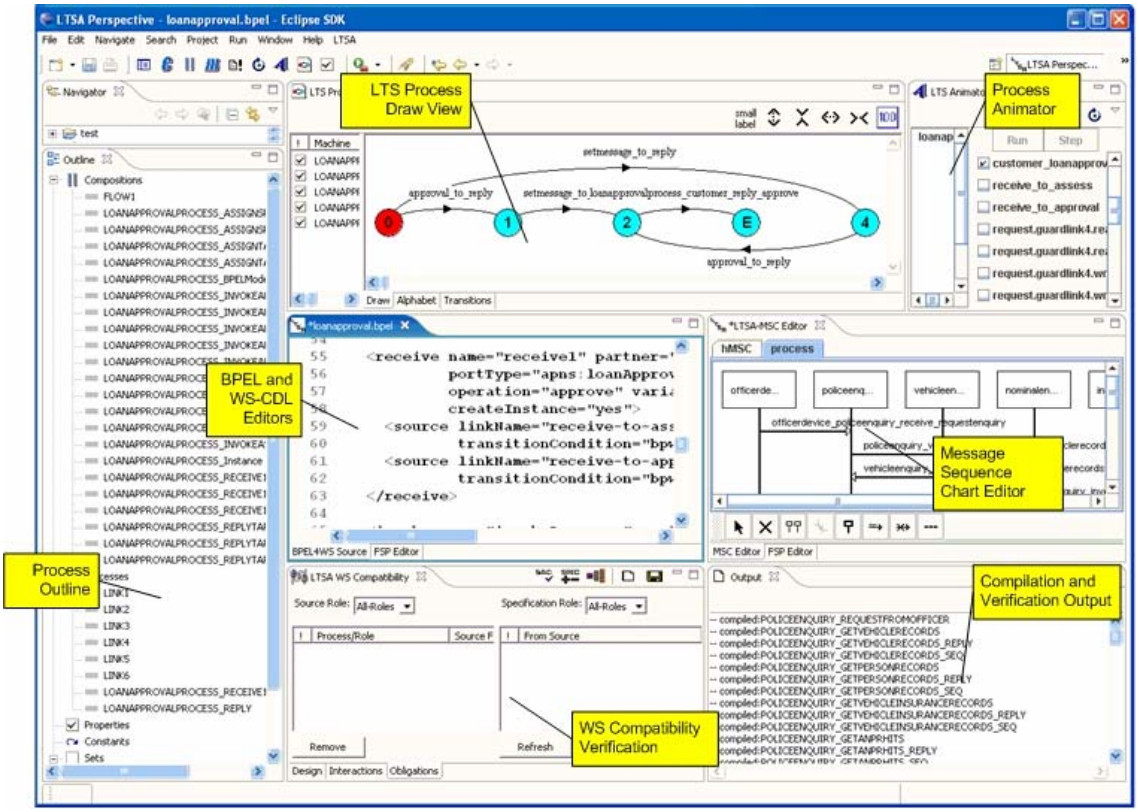
Uživatel má k dispozici všechny stavební prvky (aktivity) jazyku BPEL (plná podpora verze 2), které může jednoduše přetahovat na samotné plátno diagramu (viz

Obrázek 8). Je zde možnost vytváření vlastních komponent a jejich umístění přímo na nástrojový panel. Dále zde lze naleznout nástroje pro spouštění, simulaci procesů a mechanizmus pro odhalování chyb ap.

Jedná se o velmi silný a efektivní nástroj, bohužel vzhledem k jeho komerčnímu zaměření je pro účel této práce nevhodný.

7.2. LTSA WS-Engineer

LTSA WS-Engineer rozšiřuje program LTSA (Labelled Transition System Analyzer) a funguje jako zásuvný modul do prostředí Eclipse. Vznikl jako součást širšího projektu Web Service Engineering na Fakulty of Engineering v Londýně. Je vyvíjen vývojáři: Howard Foster, Sebastian Uchitel, Robert Chatley, Jeff Magee a Jeff Kramer. Momentálně je k dispozici (http://www.doc.ic.ac.uk/ltsa/bpel4ws/) v beta verzi 0.5.1. Podporuje práci s jazykem BPEL verze 1.1, jazykem pro choreografii služeb WS-CDL a umožňuje inženýrské modelování. Do budoucna je plánována podpora BPEL verze 2 [8].



Obrázek 9 - prostředí LTSA WS-Engineer [8]

Vzhledem k faktu, že tento nástroj je zaměřen spíše na analýzu již hotového procesu, pro samotný návrh se jeví jako značně nevýhodný. Prostředí nepůsobí uživatelsky příjemně (viz Obrázek 9) a navíc práva k tomuto programu vlastní autoři. Zdrojové kódy budou uvolněny možná někdy v budoucnosti, prozatím uživatelé nemohou měnit a redistribuovat tento nástroj [8].

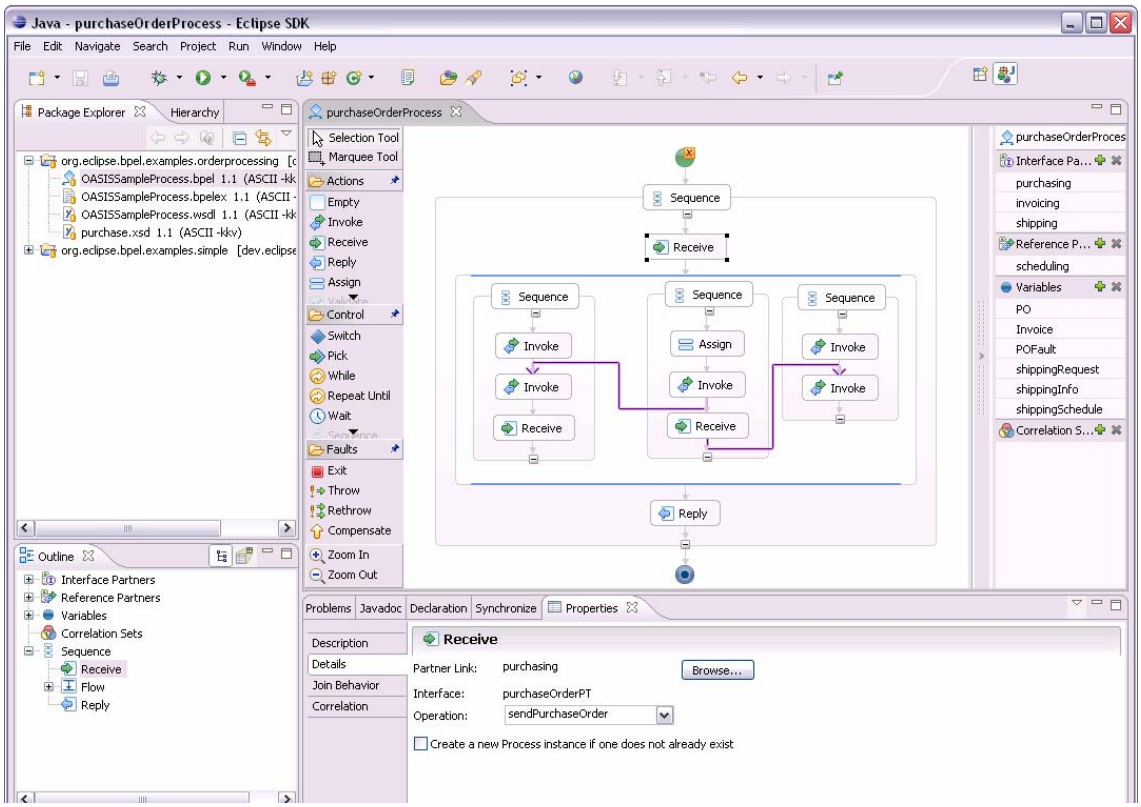
Výše uvedené důvody jasně naznačují, že se opět jedná o nástroj nevyhovující cílům této práce.

7.3. Eclipse BPEL Project

Cílem tohoto projektu je přidání komplexní podpory (grafický návrh, editace, validace, ladění atd.) jazyka BPEL verze 2 do prostředí Eclipse. Toto vše je realizováno jako zásuvný modul. http://www.eclipse.org/bpel/

Funkcionalita tohoto modulu je tvořena jednotlivými funkčními systémy [4]:

- **designér** – editor založen na GEF (Graphical Editing Framework, http://www.eclipse.org/gef/), který poskytuje grafický základ pro vytváření procesů,
- **model** – EMF (Eclipse Modeling Framework, http://www.eclipse.org/modeling/emf/) model, který zde reprezentuje specifikaci BPEL verze 2,
- **validátor** – pracuje s již zmíněným EMF modelem a produkuje chyby a varování odpovídající specifikaci,
- **spouštěcí soustava** – rozšiřitelná soustava, umožňující zpřístupnění a spouštění daných procesů z nástrojového panelu do BPEL stroje,
- **ladění** – systém, který umožní uživateli krokově procházet procesy s podporou záchytných bodů



Obrázek 10 - prostředí Eclipse BPEL Project [4]

Tento nástroj se neustále vyvíjí a poskytuje uživateli příjemné prostředí (viz Obrázek 10) podobné jako ActiveBPEL designer. Avšak narozdíl od něj se jedná open source produkt.

Z uvedených tří designérů se tento jeví jako nejvhodnější pro splnění cílů této práce.

8. POSOUZENÍ MOŽNOSTI INTEGRACE

Tato kapitola pojednává o možnosti integrace vybraných nalezených nástrojů s vybranými open source nástroji pro GIS založenými na jazyce Java.

Jak již bylo zmíněno v předchozí kapitole, pro samotnou integraci byl vybrán open source GIS nástroj uDig. Důvody vedoucí k výběru tohoto produktu byly různé. Hlavním důvodem byl fakt, že tento produkt není na našem Institutu příliš znám, natož používán a dále doporučení od vedoucího této práce. Tento krok by vedl k zvednutí povědomí o tomto produktu a poukázal na jeho možnosti.

Jediným vhodným kandidátem z uvedených nástrojů pro integraci je produkt Eclipse BPEL Project. Jeho zdrojové kódy jsou dostupné prostřednictvím CVS (Concurrent Versioning System). Obsahují kolem 900 Java tříd, spousty dalších konfiguračních souborů a vzorové příklady.

Už jenom samotný počet zmíněných tříd napovídá tomu, že pochopení kontextu nebude jednoduchou záležitostí. Dále k tomuto faktu přispívá skutečnost, že funkcionalita tohoto BPEL modulu je závislá na dalších Eclipse modulech: WPT (Web Tools Platform), EMF (Eclipse Modeling Framework), EMF (Graphical Editing Framework) a JEM (Java EMF Model Runtime driver).

Několikadenní analýza těchto zdrojových kódu (včetně zmíněných závislých modulů) se snahou o pochopení jejich souvislostí, zkompilování a samotné spuštění v prostředí Eclipse ukázala, že s největší pravděpodobností nebude reálné tuto integraci provést. Hlavním důvodem vedoucím k tomuto závěru byla složitost tohoto produktu a s ní spojený nedostatek času pro její realizaci.

9. TVORBA NÁSTROJE PRO GRAFICKÝ NÁVRH ŘETĚZENÍ SLUŽEB

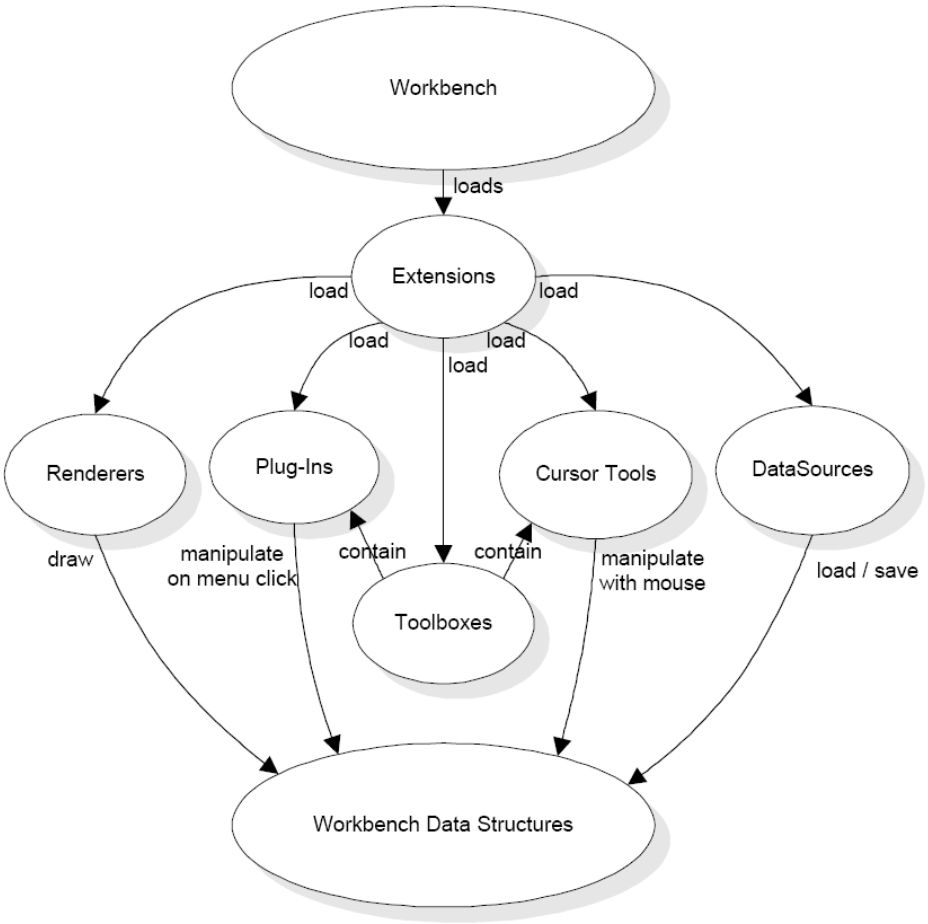
Již zmíněné okolnosti vedly k realizování druhé varianty jednoho z cílů této práce a tj. tvorby vlastního nástroje pro grafický návrh řetězení služeb, integrovaného do prostředí programového vybavení pro GIS.

V tomto případě byl zvolen pro integraci produkt OpenJUMP. Hlavní důvody vedoucí k výběru tohoto produktu byli předchozí dobré zkušenosti s tvorbou extenzí, dobrá modularita programu a jeho relativní jednoduchost.

9.1. Tvorba extenzí v prostředí OpenJUMP

OpenJUMP nabízí pro vývojáře velké množství již připravených nástrojů ulehčujících rozšiřování jeho funkcionality. Mezi ně patří dialogy a průvodci pro vkládání vstupních dat, vytahovací nabídky, XML nástroje ap. OpenJUMP je možno rozšiřovat více způsoby, viz Obrázek 11.

Po startu *„Workbench“* nahrává extenze, které mu přidávají funkcionalitu. Jedná se o soubory typu JAR, standardně uložené v podadresáři lib\ext\ v místě kde je OpenJUMP nainstalován. Tyto extenze můžou mít podobu „*plugin*“ (položky menu), *„cursor tools“* (nástrojové tlačítka), *„renderers“* (způsoby vykreslování dat) a *„datasources“* (způsoby nahrávání a ukládáni různých datových formátů) [11].



Obrázek 11 - architektura OpenJUMP [11]

Každé rozšíření musí být potomkem třídy *AbstractPlugIn*, nebo *ThreadedPlugIn*. Jak uvádí [11], *ThreadedPlugIn* poskytuje pouze lepší GUI (nejedná se o více vláknovou extenzi). Aby se daná extenze nahrála do programového prostředí, je dále nutné implementovat rozhraní *Extension*.

Při vyvíjení rozsáhlejších extenzí je výhodné spouštět OpenJUMP přímo ze zdrojových kódů. Vývojář si značně ulehčí práci, dále bude mít možnost lepšího ladění, zachytávání chyb a krokování vlastní extenze. Tato technika byla použita i v této práci.

9.2. Stručný popis tříd

K vývoji tohoto nástroje bylo použito vývojové prostředí Eclipse verze 3.2.1, OpenJUMP ve verzi 1.1.2 (jak zkompilovaná verze, tak zdrojové kódy) a Java veze 1.5.

Nástroj je tvořen 23 hlavními a 36 pomocnými třídami. Tabulka 2 uvádí stručný popis hlavních tříd. Detailnější pohled na tyto třídy viz přiložený diagram tříd.

WSDialog	Jedná se o hlavní okno aplikace, které je spouštěno jako první. Je rozděleno do čtyř částí (menu, seznam vložených služeb, stavová konzole, pracovní plocha) a jsou z něj vytvářeny instance všech hlavních tříd.
ServiceJTree	Zobrazuje a spravuje seznam vložených služeb.
DiagramBoard	Tvoří pracovní plochu a poskytuje metody pro její správu.
Console	Stavová konzole, na které je uživatel informován o probíhajících činnostech, chybách a výsledcích.
Project	Stará se o ukládání a nahrávání projektů.
OptionsDialog	Grafické rozhraní, které umožňuje nastavit některé vlastnosti aplikace.
AddServiceDialog	Umožňuje vkládání používaných webových služeb. Je možné vkládat dvěma způsoby – přímo a s podporou hledání v katalogu WSCO.
Xslt	Jak název prozrazuje, slouží k vykonávání transformací výstupu jedné služby na vstup následující s využitím jazyka XSL a XPath.
DiagramNode	Uživatelem vybrané metody, které se účastní daného procesu, jsou modelovány a zobrazovány právě pomocí této třídy.
IODialog	Grafické rozhraní, ve kterém jsou nastavovány relace a transformace mezi jednotlivými službami.
ItemsOnBoard	Manipuluje s třídami DiagramNode, které jsou umístěny na pracovní ploše.
NodesTrigger	Spouští konečný řetěz služeb a vykonává operace s tímto spojené. Proces je spouštěn v separátním vlákně.
ProgressDialog	Grafické rozhraní, informující o aktuálním vykonávání procesu s možností přerušení.
WebService	Rodičovská třída pro všechny služby. Uchovává pouze URL a při vytvoření instance zjišťuje dostupnost služby.

Wmservice	Podtřída třídy WebService, poskytující kolekci metod pro zpracování WMS služby.
WsdlService	Podtřída třídy WebService, poskytující kolekci metod pro zpracování WSDL služby.
XMLDialog	Grafické rozhraní pro práci s XML, XSL, GML soubory. Tzn. jejich zobrazení, editaci, ukládání.
GML	Podpora práce s GML soubory.
ImportDialog	Grafické rozhraní, které umožňuje uživateli importovat procesy uložené v jazyku BPEL.
ExportWizardPanel1	První část průvodce exportem řetězu služeb do jazyka BPEL.
ExportWizardPanel2	Druhá část průvodce, spolu s prováděním samotného exportu.
WSExtension	Implementace rozhraní Extension, zabezpečující nahrání extenze do prostředí OpenJUMP.
WSPlugin	Podtřída třídy AbstractPlugIn informující o tom, že tato aplikace je přídavným modulem.

Tabulka 2 - stručný popis hlavních tříd extenze

Zmíněné třídy spolupracují s třídami pomocnými. Většina těchto tříd je využívaná při exportu do jazyka BPEL, kde tyto třídy tvoří jeho jednotlivé elementy. Původní představa byla taková, že při exportu bude využita nějaká již hotová volně dostupná Java knihovna, která již bude umět s tímto jazykem pracovat:

- **BPEL4WS –** API od firmy IBM s dobrou dokumentací. Relativně jednoduché použití. Bohužel podpora BPEL specifikace pouze verze 1.1. Tento fakt vedl k některým nepřekonatelným problémům, a tudíž bylo její použití zamítnuto;
- **Eclipse BPEL Project** – Knihovny tohoto produktu mají návaznost na další Eclipse moduly (jako EMF, GEF ap.). Co se týče vývojářské části dokumentace je prozatím velmi omezená. Vzhledem k rozsahu všech

těchto modulů, stavu dokumentace a s tím spojené časové náročnosti se nepodařilo korektně tyto knihovny použít.

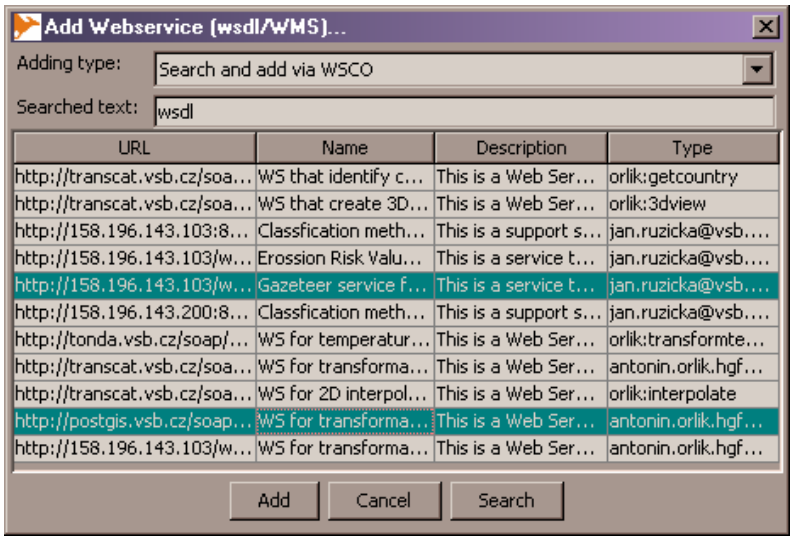
9.3. Popis GUI nástroje

Nejefektivnější cestou jak demonstrovat uživatelské prostředí a možnosti tohoto nástroje je přímo na praktickém příkladě. Jedná se o situaci, která již byla popsána v kapitole 5.4.

9.3.1. Přidávání služeb

Tato volba se nachází v menu *Tools*. Uživatel má možnost přidávat služby, které bude zapojovat do řetězu dvěma způsoby:

- **přímo –** Vložením URI dané služby;
- **prostřednictvím WSCO katalogu** – S využitím služby, kterou poskytuje tento katalog. Uživatel si může dohledat zadáním klíčového slova služby, které chce používat a najednou je přidat (viz Obrázek 12).

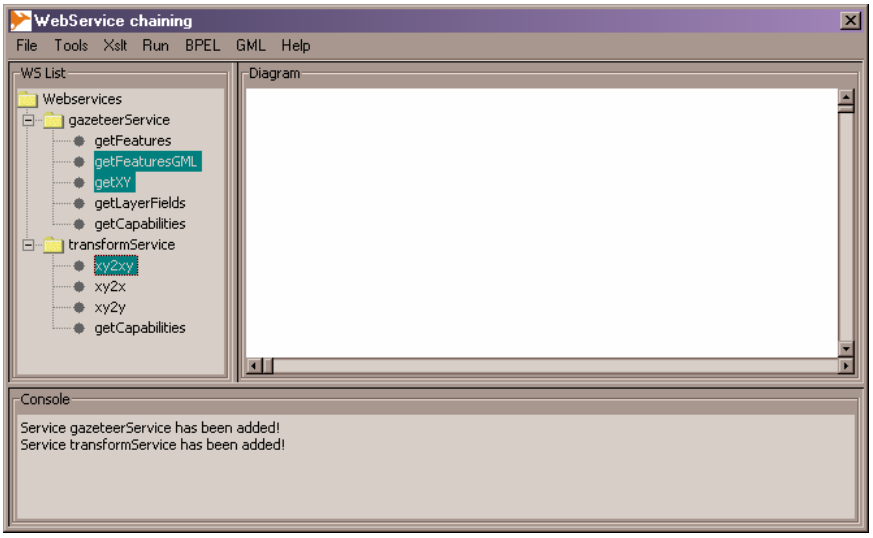


Obrázek 12 - GUI, přidávaní služeb s vyhledáváním ve WSCO

Tímto způsobem lze přidávat služby typu WSDL i WMS. Na obrázku níže jsou už vidět dvě přidané služby, které tento ukázkový příklad používá. Jedná se o služby:

- http://158.196.143.103/wsdl/gazeteer.wsdl - pomocí této služby získá uživatel souřadnice zadané obce (metoda *getXY*) a obdrží GML vrstvu s vodními plochami (metoda *getFeaturesGML*),

- http://postgis.vsb.cz/soap/wsdl/transform_coord.wsdl - prostřednictvím metody *xy2xy* převede souřadnice z jednoho souřadného systému do druhého



Obrázek 13 - GUI, přidané služby

9.3.2. Vytváření řetězu služeb

V dalším kroku uživatel přesune pomocí operace *„Drag-and-Drop“* jednotlivé metody na pracovní plochu v pořadí v jakém budou spouštěny. Při každé takovéto akci je mu nabídnuto grafické rozhraní specifikující vstupy dané metody a jejich závislost s předchozím výstupem. V tomto okně lze nastavit čtyři druhy propojení:

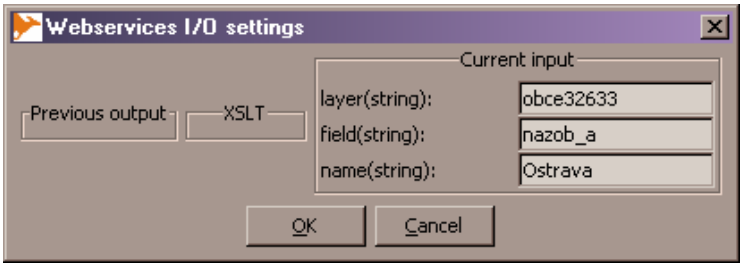
- **bez vazby (*user defined)*** – uživatel vloží přímo konkrétní hodnotu aktuálního vstupu,

- **s přímou vazbou** – předchozí výstup je bez jakékoliv změny předán na aktuální vstup,

- **s úpravou pomocí XPath výrazu (*XPath expression)*** – pomocí jazyka XPath lze zapsat jednoduchý transformační výraz,

- **s využitím XSL transformace** – umožňuje provádět více náročné transformace mezi výstupy a vstupy s využitím možností jazyka XSL (tato transformace se připojuje k řetězu v menu *Xslt*)

Vzhledem k tomu, že metoda *getXY* stojí v tomto řetězu na prvním místě, lze vložit její vstupní hodnoty pouze přímo (viz Obrázek 14). Vrací souřadnice zadané obce v textovém tvaru „POINT(X Y)“;

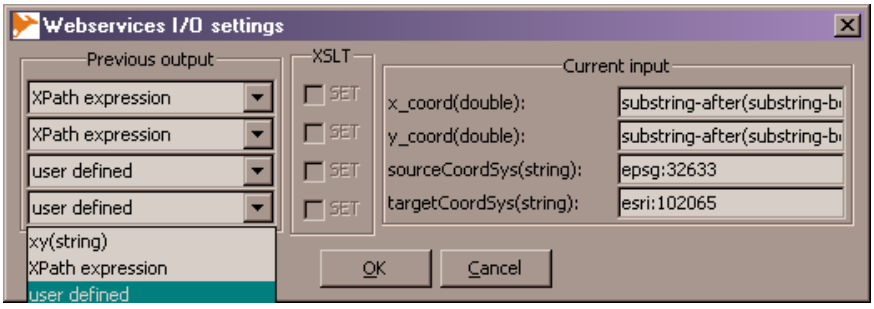


Obrázek 14 - GUI, přidání medody getXY

Druhá metoda *xy2xy* převádí souřadnice mezi různými formáty (zde z UTM formátu do S-JTSK). Tato metoda vyžaduje na vstupu souřadnice X, Y a dále kódové označení zdrojového a cílového souřadnicového systému (viz Obrázek 15). Předchozí metoda nevrací souřadnice ve vhodném tvaru, proto je nutné ho upravit. V tomto případě je využit XPath výraz (XSL transformace viz příloha 14.2):

- *substring-after(substring-before($getXYResponse.xy,' '),'(')* – vybere z předchozího výstupu „POINT(X Y)“ souřadnici X,
- *substring-after(substring-before($getXYResponse.xy,')'),' ')* – naopak získá souřadnici Y

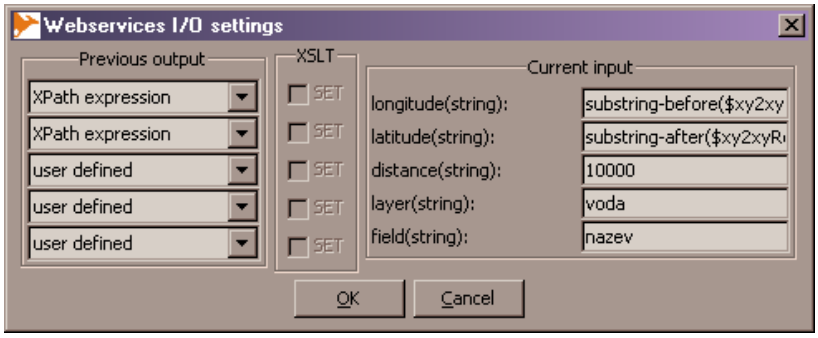
Tato metoda vrací souřadnice jako řetězec ve tvaru „X,Y“.



Obrázek 15 - GUI, přidání metody xy2xy

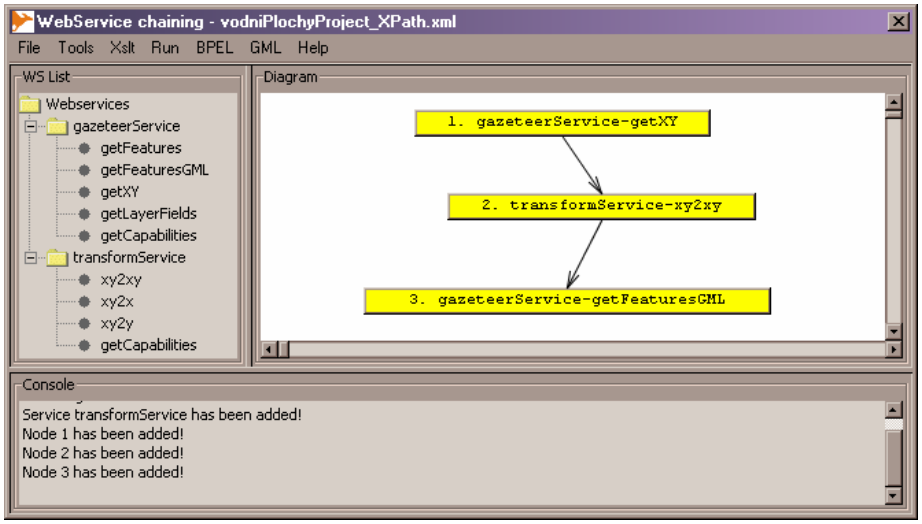
Poslední metodou v tomto procesu je metoda *getFeaturesGML*, která vrací vodní plochy (GML vrstvu) v určité vzdálenosti od zadaného bodu (viz Obrázek 16). Zde bylo opět zapotřebí upravit předchozí výstup (XSL transformace viz příloha 14.2):

- *substring-before($xy2xyResponse.xy,',')* – získá souřadnici X
- *substring-after($xy2xyResponse.xy,',')* – získá souřadnici Y



Obrázek 16 - GUI, přidání metody getFeaturesGML

Jednotlivé metody jsou zobrazeny na pracovní ploše v místě, na kterém byly vloženy (operace *„Drop“*). Obrázek 17 znázorňuje již všechny tři vložené metody. Uživatel má dále možnost tyto uzly (metody na ploše) libovolně rozmísťovat, případně opětovně upravovat jejich vzájemné závislosti.

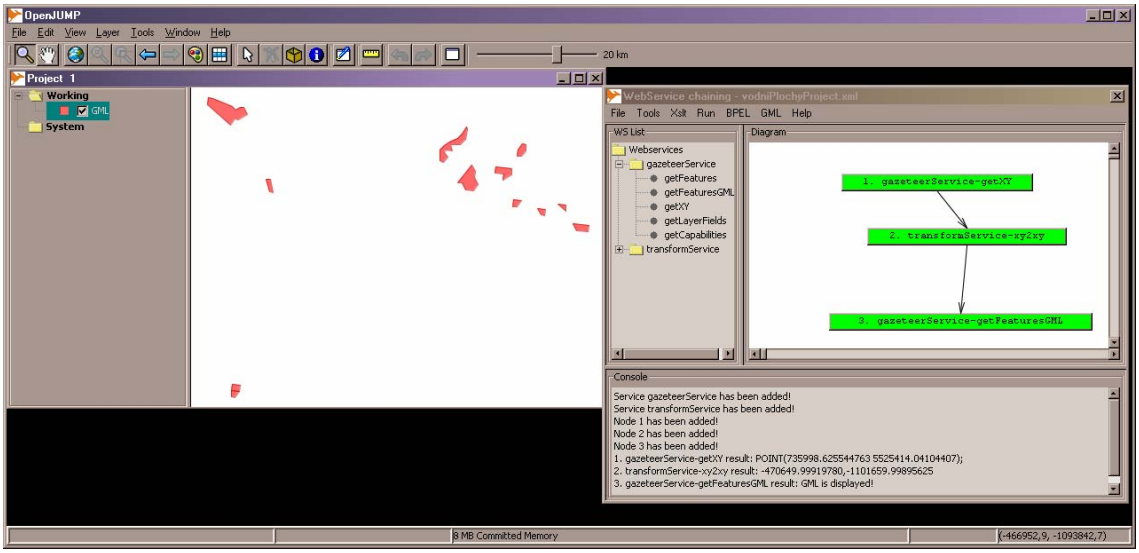


Obrázek 17 - GUI, vytvořený řetěz služeb

9.3.3. Spouštění řetězu služeb

Z menu *Run* lze vyvolat spuštění aktuálního řetězu služeb. Během provádění je uživatel informován o jeho stavu a mezivýsledcích jednotlivých uzlů (metod). Při výskytu nějaké chyby je běh zastaven a uzel, kde chyba nastala zvýrazněn červeně.

Samozřejmě pokud se jedná o extenzi známou chybu, uživatel je o ní informován. V případě potřeby lze ručně tento běh přerušit v jakémkoliv místě. Aby byl OpenJUMP schopný zobrazit GML vrstvu, potřebuje mít k dispozici ještě šablonu, která mu sdělí podrobnosti o geometrii a atributech dané vrstvy. Tuto šablonu automaticky generuje extenze a výsledné GML ihned zobrazí. Na obrázku níže je vidět výsledek ukázkového procesu.



Obrázek 18 - GUI, výsledek ukázkového procesu

Uživatel může daný řetěz služeb opakovaně spouštět a dynamicky měnit hodnoty vstupů jednotlivých metod dle jeho požadavků.

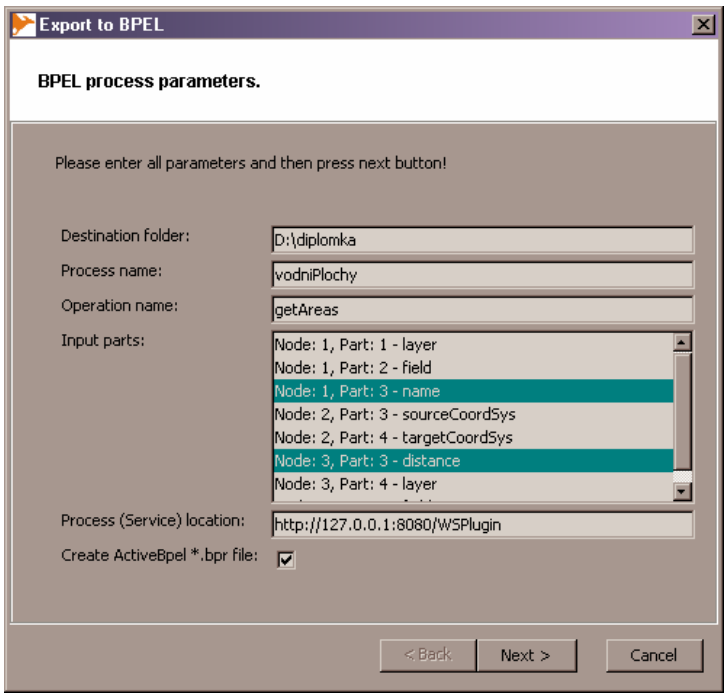
9.3.4. Export do jazyka BPEL

V případě, že by chtěl uživatel daný řetěz služeb poskytovat širší veřejnosti, má možnost ho exportovat jako proces do jazyka BPEL. Pod jakým strojem bude tento proces provozován je zcela na uživateli, ale vzhledem k velmi dobrým zkušenostem s ActiveBPEL engine, lze exportovat přímo do tohoto formátu (vygenerování všech souborů potřebných pro běh pod tímto strojem).

Obrázek 19 znázorňuje formulář pro export se základními parametry:

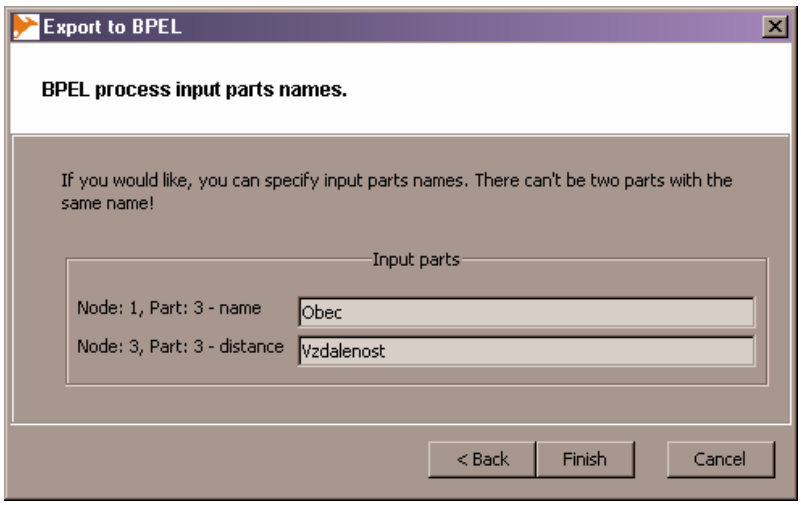
- **název procesu (Process name)**
- **jméno metody (Operation name)** – metoda pod kterou bude daný proces přístupný (proces se jeví navenek jako standardní webová služba),

- **specifikace vstupu (Input Parts)** – jak je patrné z 9.3.2, většina metod potřebuje ke své správné funkci ještě různé podpůrné parametry, které jsou pro konečného uživatele relevantní. V tomto případě jsou důležité pouze dva parametry, a to název obce a vzdálenost,
- **lokace služby** – místo, kde bude daná služba vystavena,
- **soubor ve formátu ActiveBPEL**



Obrázek 19 - GUI, export do BPEL, část 1.

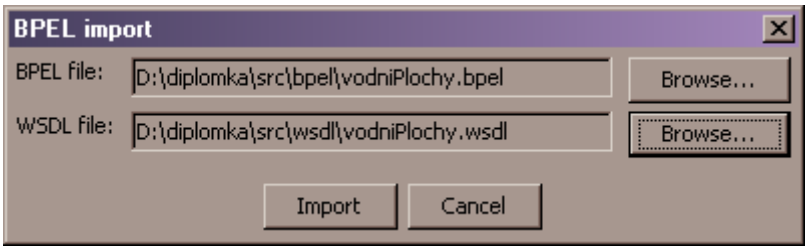
V druhém kroku exportu může (musí v případě výskytů duplicitních názvů) uživatel ještě upřesnit jména vstupních parametrů (viz Obrázek 20).



Obrázek 20 - GUI, export do BPEL, část 2.

9.3.5. Import z jazyka BPEL

Extenze umožňuje import z jazyka BPEL, ovšem s omezením pouze na soubory které byly vytvořeny touto aplikací. Je tomu tak, protože jazyk BPEL podporuje vytváření mnohem komplexnějších procesů a tato aplikace podporuje pouze sekvenční procesy.



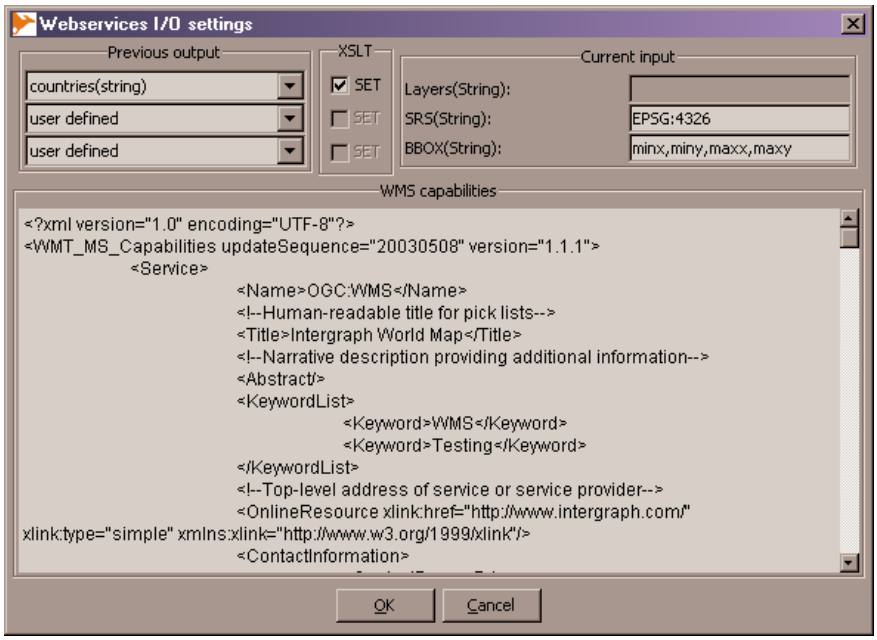
Obrázek 21 - GUI, okno pro import BPEL

Vzhledem k tomu, že BPEL jazyk neobsahuje elementy pro uchovávání grafické složky procesu, načtené metody (uzly) jsou zarovnány pravidelně k začátku pracovní plochy.

9.3.6. Zapojení WMS služeb do řetězu

V současnosti je na Internetu k dispozici spousta volně přístupných WMS služeb, které nám poskytují mapové podklady nejrůznějšího druhu. Využití těchto služeb zapojených do řetězu je široké. Nejčastější je však případ, kdy výsledek řetězu bude nějaká zájmová oblast. Tímto typem služby by bylo možné např. nahradit službu v ukázkovém procesu (viz 5.4), která vrací GML vrstvu vodních ploch. Ovšem museli bychom ještě konkrétní WMS službě předřadit službu, která by vygenerovala ze souřadnic dané obce a vzdálenosti ohraničující obálku požadované oblasti.

Stejně jako u WSDL služeb, lze provázat její vstupy s výstupem předchozí služby (viz Obrázek 22).



Obrázek 22 - GUI, okno pro přidávání WMS služby

Jazyk BPEL nepodporuje služby WMS, vytvořené řetězy obsahující tento typ služby nelze do něj exportovat.

Vstup každé WMS služby je zde tvořen třemi parametry:

- **vrstvy (Layers)** – čárkou oddělený seznam vrstev, které chce uživatel zobrazit,
- **prostorový referenční systém (SRS)** – identifikátor prostorového referenčního systému v kterém má být požadovaná mapa vrácena,
- **ohraničující obálka (BBOX)** – jedná se o nastavení dolní levé a horní pravé hranice v jednotkách příslušejících danému SRS

Po spuštění takovéhoto řetězu služeb, který obsahuje WMS službu, je výsledek okamžitě zobrazen v prostředí OpenJUMP. V případě, že uživatel nenastaví BBOX, použije se hodnota, kterou má aktuálně nastaveno zobrazovací okno OpenJUMPu.

9.3.7. Některé další funkce extenze

Tato extenze má zabudované některé další funkce, jako:

- ukládaní a nahrávání projektů,

- mazaní jednotlivých služeb,
- mazání uzlů (metod) na pracovní ploše,
- restartovaní celého prostředí extenze,
- zapamatování si některých nastavení

10. ZÁVĚR

Cílem diplomové práce bylo prozkoumat stávající možnosti řetězení webových služeb a následně je implementovat do prostředí open source GIS.

V současné době není pro nás tak důležité řetězení webových služeb na úrovni choreografie, nejdříve je zapotřebí zvládnout problematiku orchestrace. Právě jazyk BPEL se jeví jako vhodný pro pokrytí všech jejich potřeb. Už teď má dobrou aplikační podporu jak ze strany komerční, tak ze strany open source. Ovšem ne vždy je tato podpora dostatečná a je už jenom na uživateli, který produkt zvolí.

Vzhledem k požadavkům této práce, byly nalezeny tři produkty pro grafický návrh řetězení webových služeb, naznačující možnost integrace do prostředí open source GIS (konkrétně programu uDig). Jejich analýza však prokázala, že vhodným kandidátem je pouze produkt Eclipse BPEL Project. Bohužel jeho složitost vedla k přistoupení k druhé variantě – k tvorbě vlastní aplikace.

Tato aplikace byla vytvořena jako zásuvný modul do produktu OpenJUMP. V současnosti umožňuje grafický návrh sekvenčního řetězení webových služeb s podporou přímého zobrazování mapových výstupů v OpenJUMP. Na rozdíl od všech ostatních produktů určených k řetězení webových služeb, umožňuje do řetězu zařazovat kromě WSDL služeb i služby WMS. Jazyky XSL a XPath se zde zabezpečují transformace výstupu jednotlivých metod (služeb).

Extenze mimo jiné poskytuje schopnost jednoduchého exportu (importu) do (z) jazyka BPEL a tím umožňuje ze složitých řetězů služeb vytvářet business procesy, které se jeví navenek jako jednoduché služby vhodné distribuce k dalším uživatelům. K tomu všemu kladně pomáhá podpora open source stroje ActiveBPEL engine, která dovoluje okamžité používání vytvořeného procesu. Všechny tyto dovednosti byly demonstrovány na praktickém ukázkovém řetězu služeb.

Vzhledem k nedostupnosti praktických řešení v této oblasti, přispívá tato práce pozitivně k jejímu rozvoji a celkově k rozvoji servisně orientované architektury, která bude pak dobrým základem pro vytvoření nového business modelu. Potom už bude reálné zabývat se samotnou choreografií webových služeb.

11. POUŽITÁ LITERATÚRA

1. Active Endpoints, Inc. [online]. 2006. [cit. 2007-03]. Dostupný na WWW: http://www.active-endpoints.com/

2. BLANVALVET, S; BOLIE, J; CARDELLA, M; CAREY, S; CHANDRAN, P; COENE, Y; GEMINIUC, K; JURIČ, M; NGUYEN, H; PODUVAL, A; PRAVIN, L; THOMAS, J; TODD, D. *BPEL Cookbook: Best Practivec for SOA-based integration and composite applications development.* Birmingham: Packt Publishing Ltd., 2006. ISBN 1-904811-33-7

3. DĚRGEL, P; ŠELIGA, M. Metainformation systems and Web services. [online]. 2004. [cit. 2007-01]. Dostupný na WWW: http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2004/Sbornik/Referaty/seliga.htm.

4. Eclipse. [online]. 2006. [cit. 2007-03]. Dostupný na WWW: http://www.eclipse.org/

5. HAWLITZEK; F. *Java 2 příručka programátora.* Praha Grada Publishing, spol. s r.o. 2002. ISBN 80-247-9060-2

6. HOUSPANOSSIAN, A.Enhancing a BPEL4WS Engine – Supporting the Execution of Flexible WS-flows According to the ReFFlow Model. [online]. 2006. [cit. 2007-02]. Dostupný na WWW: http://www.dvs1.informatik.tu-darmstadt.de/publications/BScs/Alejandro.Houspanossian_Apr06.pdf

7. KOSEK, J. Inteligentní podpora navigace na WWW s využitím XML. *Diplomová práce.* [online]. 2002. [cit. 2007-01]. Praha. Dostupný na WWW: http://www.kosek.cz/diplomka/dp.pdf

8. LTSA WS-Engineer. [online]. 2006. [cit. 2007-02]. Dostupný na WWW: http://www.doc.ic.ac.uk/ltsa/bpel4ws/

9. OASIS. [online]. 2006. [cit. 2007-02]. Dostupný na WWW: http://www.oasis-open.org/

10. OpenGIS. [online]. 2007. [cit. 2007-03]. Dostupný na WWW: http://www.opengeospatial.org/

11. OpenJUMP. [online]. 2007. [cit. 2007-03]. Dostupný na WWW: http://openjump.org/

12. RUŽIČKA, J; MARŠÍK, V; ŠELIGA, M; PRAGER, M; ORLÍK, A; HORÁKOVÁ, B. WS-CDL for Geoinformatics. [online]. 2006. [cit. 2007-01]. Dostupný na WWW: http://gisak.vsb.cz/wsco/publikace/RuzickaPaper.pdf

13. SAYAR, A; PIERCE, M; FOX, G. OGC Compatible Geographical Information Systems Web Services. [online]. 2005. [cit. 2007-01]. Dostupný na WWW: http://grids.ucs.indiana.edu/ptliupages/publications/ogctech_report.pdf.

14. SEGNER, M. Web Services [online]. 2004. [cit. 2007-03]. Dostupný na WWW: http://www.mygrid.org.uk/wiki/Mygrid/PresentationStore

15. SCHITTKO, C. Web Services Orchestration with BPEL. [online]. 2006. [cit. 2007-02]. Dostupný na WWW: http://www.idealliance.org/proceedings/xml03/slides/schittko/schittko.ppt

16. TANG W. SELWOOD J. Connecting our World – GIS Web Services. ESRI Press. 2003. ISBN 1589480759

17. THAN V. Web Service Orchestration - An open and standardised approach for creating advanced services. [online]. 2003. [cit. 2007-01]. Dostupný na WWW: http://www.eurescom.de/message/messageJun2003/Web_Service_Orchestration.asp

18. uDig. [online]. 2006. [cit. 2007-03]. Dostupný na WWW: http://udig.refractions.net/confluence/display/UDIG/Home

19. W3C. [online]. 2006. [cit. 2007-03]. Dostupný na WWW: http://www.w3.org/

12. SEZNAM OBRÁZKŮ

13. SEZNAM TABULEK

14. SEZNAM PŘÍLOH

14.1. Příloha 1 – Ukázkový BPEL proces

- BPEL soubor (vodniPlochy.bpel):

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        exitOnStandardFault="yes"
        name="vodniPlochy" suppressJoinFailure="yes"
        targetNamespace="cz.vsb.gisak.parek.vodniPlochy"
        xmlns:ns1="http://tonda.vsb.cz/gazeteer"
        xmlns:ns2="http://postgis.vsb.cz/transform"
        xmlns:ns3="cz.vsb.gisak.parek">
        <partnerLinks>
            <partnerLink myRole="vodniPlochy" name="getAreasLink"
partnerLinkType="ns3:vodniPlochyPLT"/>
            <partnerLink name="gazeteerLink" partnerLinkType="ns3:gazeteerPLT"
partnerRole="gazeteerRole"/>
            <partnerLink name="xy2xyLink" partnerLinkType="ns3:xy2xyPLT"
partnerRole="xy2xy"/>
        </partnerLinks>
        <variables>
            <variable messageType="ns3:getAreasRequest" name="getAreasRequest"/>
            <variable messageType="ns3:getAreasResponse" name="getAreasResponse" />
            <variable messageType="ns1:getXYRequest" name="getXYRequest" />
            <variable messageType="ns1:getXYResponse" name="getXYResponse" />
            <variable messageType="ns2:xy2xyRequest" name="xy2xyRequest" />
            <variable messageType="ns2:xy2xyResponse" name="xy2xyResponse" />
            <variable messageType="ns1:getFeaturesGMLRequest"
name="getFeaturesGMLRequest" />
            <variable messageType="ns1:getFeaturesGMLResponse"
name="getFeaturesGMLResponse" />
        </variables>
        <faultHandlers>
              <catchAll>
                 <sequence>
                    <compensate name="ReverseCompletedTransactions"/>
                    <assign name="GenerateFaultMessage">
                       <copy>
                          <from>
                             <literal>An error occurred while submitting the order. All
transactions have been canceled.</literal>
                          </from>
                          <to part="result" variable="getAreasResponse"/>
                       </copy>
                    </assign>
                    <reply name="SendError" operation="getAreas"
partnerLink="getAreasLink" portType="ns3:vodniPlochyPT" variable="getAreasResponse"/>
                 </sequence>
              </catchAll>
        </faultHandlers>
        <sequence>
            <receive createInstance="yes" operation="getAreas"
partnerLink="getAreasLink" portType="ns3:vodniPlochyPT" variable="getAreasRequest"/>
               <assign>
                   <copy>
                       <from>$getAreasRequest.Obec</from>
                       <to part="name" variable="getXYRequest"/>
                   </copy>
                   <copy>
                       <from>'obce32633'</from>
                       <to part="layer" variable="getXYRequest"/>
                   </copy>
                   <copy>
                       <from>'nazob_a'</from>
                       <to part="field" variable="getXYRequest"/>
                   </copy>
               </assign>
```

```
               <invoke inputVariable="getXYRequest" name="InvokeGetXY" operation="getXY"
outputVariable="getXYResponse" partnerLink="gazeteerLink" portType="ns1:gazeteerPort"/>
                <assign>
                     <copy>
                         <from>substring-after(substring-before($getXYResponse.xy,'
'),'(')</from>
                         <to part="x_coord" variable="xy2xyRequest"/>
                     </copy>
                     <copy>
                         <from>substring-after(substring-before($getXYResponse.xy,')'),'
')</from>
                         <to part="y_coord" variable="xy2xyRequest"/>
                     </copy>
                     <copy>
                         <from>'epsg:32633'</from>
                         <to part="sourceCoordSys" variable="xy2xyRequest"/>
                     </copy>
                     <copy>
                         <from>'esri:102065'</from>
                         <to part="targetCoordSys" variable="xy2xyRequest"/>
                     </copy>
                </assign>
                <invoke inputVariable="xy2xyRequest" name="Invokexy2xy" operation="xy2xy"
outputVariable="xy2xyResponse" partnerLink="xy2xyLink" portType="ns2:transformPort"/>
                <assign>
                     <copy>
                         <from>substring-before($xy2xyResponse.xy,',')</from>
                         <to part="longitude" variable="getFeaturesGMLRequest"/>
                     </copy>
                     <copy>
                         <from>substring-after($xy2xyResponse.xy,',')</from>
                         <to part="latitude" variable="getFeaturesGMLRequest"/>
                     </copy>
                     <copy>
                         <from>$getAreasRequest.Vzdalenost</from>
                         <to part="distance" variable="getFeaturesGMLRequest"/>
                     </copy>
                     <copy>
                         <from>'voda'</from>
                         <to part="layer" variable="getFeaturesGMLRequest"/>
                     </copy>
                     <copy>
                         <from>'nazev'</from>
                         <to part="field" variable="getFeaturesGMLRequest"/>
                     </copy>
                </assign>
                <invoke inputVariable="getFeaturesGMLRequest" name="InvokeGetGML"
operation="getFeaturesGML" outputVariable="getFeaturesGMLResponse"
partnerLink="gazeteerLink" portType="ns1:gazeteerPort"/>
                <assign>
                    <copy>
                        <from>$getFeaturesGMLResponse.countries</from>
                        <to part="result" variable="getAreasResponse"/>
                    </copy>
               </assign>
               <reply operation="getAreas" partnerLink="getAreasLink"
portType="ns3:vodniPlochyPT" variable="getAreasResponse"/>
        </sequence>
</process>
```

- WSDL soubor (vodniPlochy.wsdl):

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="vodniPlochy"
    targetNamespace="cz.vsb.gisak.parek"
    xmlns:tns="cz.vsb.gisak.parek"
    xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:ns1="http://tonda.vsb.cz/gazeteer"
    xmlns:ns2="http://postgis.vsb.cz/transform">
```

```
  <wsdl:import namespace="http://tonda.vsb.cz/gazeteer"
location="http://158.196.143.103/wsdl/gazeteer.wsdl"/>
  <wsdl:import namespace="http://postgis.vsb.cz/transform"
location="http://postgis.vsb.cz/soap/wsdl/transform_coord.wsdl"/>
  <wsdl:message name="getAreasRequest">
    <wsdl:part name="Obec" type="xsd:string"/>
    <wsdl:part name="Vzdalenost" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="getAreasResponse">
    <wsdl:part name="result" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="vodniPlochyPT">
    <wsdl:operation name="getAreas">
      <wsdl:input name="getAreasRequest" message="tns:getAreasRequest"/>
      <wsdl:output name="getAreasResponse" message="tns:getAreasResponse"/>
    </wsdl:operation>
  </wsdl:portType>
<plnk:partnerLinkType name="vodniPlochyPLT">
   <plnk:role name="vodniPlochy">
      <plnk:portType name="tns:vodniPlochyPT"/>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="gazeteerPLT">
   <plnk:role name="gazeteerRole">
      <plnk:portType name="ns1:gazeteerPort"/>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="xy2xyPLT">
   <plnk:role name="xy2xy">
      <plnk:portType name="ns2:transformPort"/>
   </plnk:role>
</plnk:partnerLinkType>
<wsdl:binding name="vodniPlochyServiceSoapBinding" type="tns:vodniPlochyPT">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    <wsdl:operation name="getAreas">
       <wsdlsoap:operation soapAction=""
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"/>
         <wsdl:input name="getAreasRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="cz.vsb.gisak.parek" use="encoded"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"/>
         </wsdl:input>
         <wsdl:output name="getAreasResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="cz.vsb.gisak.parek" use="encoded"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"/>
         </wsdl:output>
       </wsdl:operation>
</wsdl:binding>
<wsdl:service name="vodniPlochyService">
    <wsdl:port name="vodniPlochyPT" binding="tns:vodniPlochyServiceSoapBinding">
      <wsdlsoap:address location="http://localhost:8080/active-bpel/vodniPlochy"
         xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

- PDD (process deployment descriptor pro ActiveBPEL engine) soubor (vodniPlochy.pdd):

```
<?xml version="1.0" encoding="UTF-8"?>
<process location="bpel/vodniPlochy.bpel"
    name="bpelns:vodniPlochy"
    xmlns="http://schemas.active-endpoints.com/pdd/2005/09/pdd.xsd"
    xmlns:bpelns="cz.vsb.gisak.parek.vodniPlochy"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
     <partnerLinks>
       <partnerLink name="gazeteerLink">
         <partnerRole endpointReference="static">
            <wsa:EndpointReference xmlns:s="http://tonda.vsb.cz/gazeteer"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
               <wsa:Address>http://158.196.143.103/cgi-bin/gazeteer.cgi</wsa:Address>
               <wsa:PortType>s:gazeteerPort</wsa:PortType>
```

```
               <wsa:ServiceName
PortName="gazeteerPort">s:gazeteerService</wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
      <partnerLink name="xy2xyLink">
         <partnerRole endpointReference="static">
            <wsa:EndpointReference xmlns:s="http://postgis.vsb.cz/transform"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
               <wsa:Address>http://postgis.vsb.cz/cgi-
bin/transform_coord.cgi</wsa:Address>
               <wsa:PortType>s:transformPort</wsa:PortType>
              <wsa:ServiceName
PortName="transformPort">s:transformService</wsa:ServiceName>
            </wsa:EndpointReference>
         </partnerRole>
      </partnerLink>
      <partnerLink name="getAreasLink">
         <myRole allowedRoles="" binding="RPC" service="vodniPlochyService"/>
      </partnerLink>
     </partnerLinks>
    <wsdlReferences>
      <wsdl namespace="cz.vsb.gisak.parek" location="wsdl/vodniPlochy.wsdl"/>
      <wsdl namespace="http://tonda.vsb.cz/gazeteer"
location="http://158.196.143.103/wsdl/gazeteer.wsdl"/>
      <wsdl namespace="http://postgis.vsb.cz/transform"
location="http://postgis.vsb.cz/soap/wsdl/transform_coord.wsdl"/>
   </wsdlReferences>
</process>
```

- XML (WSDL Katalog) soubor (wsdlCatalog.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdlCatalog>
   <wsdlEntry location="wsdl/vodniPlochy.wsdl" classpath="wsdl/vodniPlochy.wsdl" />
</wsdlCatalog>
```

14.2. Příloha 2 – transformace pomocí XSL

- Automaticky vygenerovaný vstupní XML řetězec (pro ukázkový proces). Výraz *PREVIEW* je během spuštění řetězu nahrazován hodnotami výstupů konkrétních metod:

```
<?xml version="1.0" encoding="utf-8"?>
<DiagramBoard>
	<Node>
		<id>2</id>
		<type>wsdl</type>
		<input>
			<part>
				<idPart>0</idPart>
				<relatedNode>1</relatedNode>
				<relatedPartType>string</relatedPartType>
				<relatedValue>PREVIEW</relatedValue>
			</part>
			<part>
				<idPart>1</idPart>
				<relatedNode>1</relatedNode>
				<relatedPartType>string</relatedPartType>
				<relatedValue>PREVIEW</relatedValue>
			</part>
		</input>
	</Node>
	<Node>
		<id>3</id>
		<type>wsdl</type>
		<input>
			<part>
				<idPart>0</idPart>
				<relatedNode>2</relatedNode>
				<relatedPartType>string</relatedPartType>
```

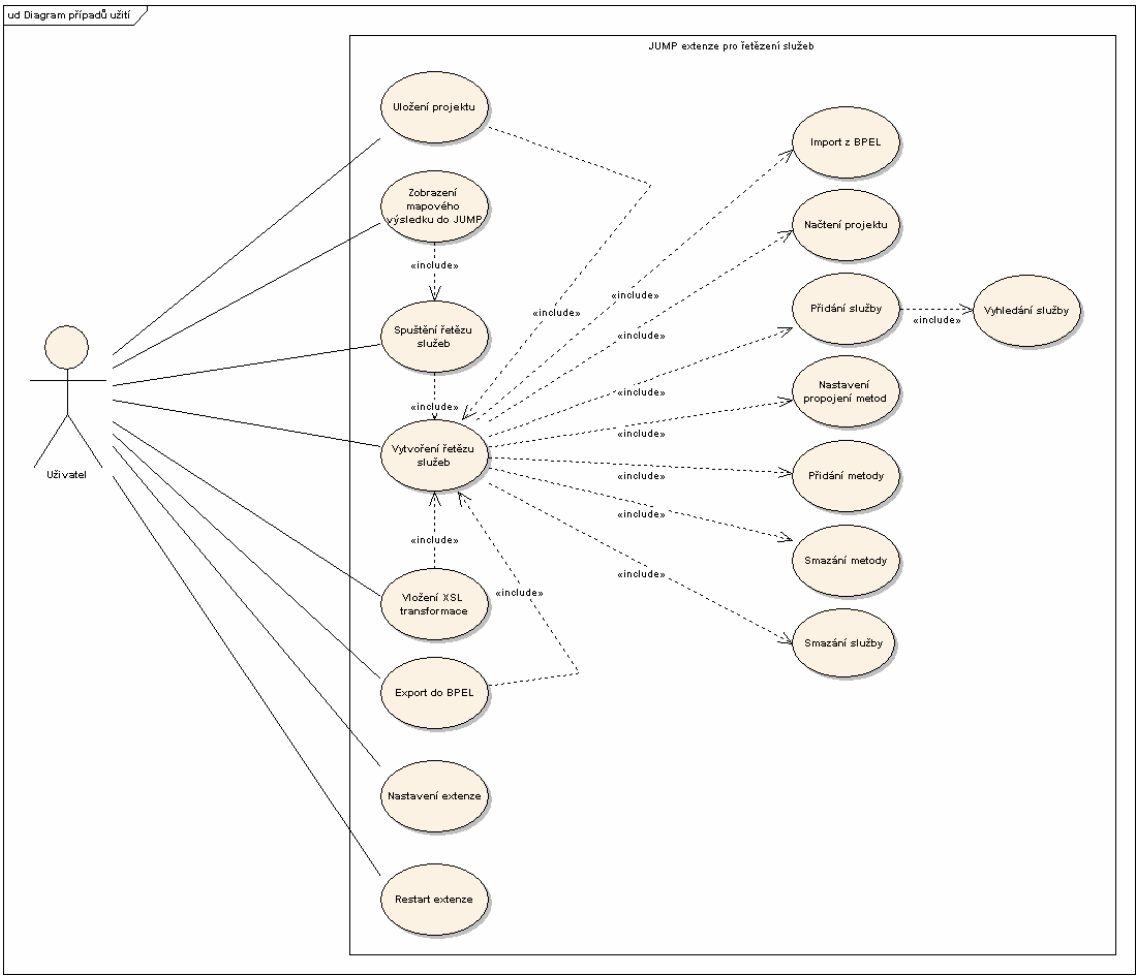
```
                        <relatedValue>PREVIEW</relatedValue>
                    </part>
                    <part>
                        <idPart>1</idPart>
                        <relatedNode>2</relatedNode>
                        <relatedPartType>string</relatedPartType>
                        <relatedValue>PREVIEW</relatedValue>
                    </part>
                </input>
        </Node>
</DiagramBoard>
```

- Transformační XSL soubor napsaný uživatelem (pro ukázkový proces):

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

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

    <xsl:template match="DiagramBoard/Node">
        <xsl:if test="input/part/relatedValue!='PREVIEW'">
            <xsl:copy><xsl:apply-templates/></xsl:copy>
        </xsl:if>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/id">
        <xsl:copy><xsl:apply-templates/></xsl:copy>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/type">
        <xsl:copy><xsl:apply-templates/></xsl:copy>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/input">
        <xsl:copy><xsl:apply-templates/></xsl:copy>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/input/part">
        <xsl:if test="relatedValue!='PREVIEW'">
             <xsl:copy><xsl:apply-templates/></xsl:copy>
        </xsl:if>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/input/part/idPart">
        <xsl:copy><xsl:apply-templates/></xsl:copy>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/input/part/relatedNode">
        <xsl:copy><xsl:apply-templates/></xsl:copy>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/input/part/relatedPartType">
        <xsl:copy><xsl:apply-templates/></xsl:copy>
    </xsl:template>

    <xsl:template match="DiagramBoard/Node/input/part/relatedValue">
        <xsl:choose>

            <xsl:when test="preceding-sibling::idPart='0' and ancestor::Node[id='2']">
                <xsl:element name="relatedValue">
                       <xsl:value-of select="substring-after(substring-before(.,'
'),'(')"/>
                </xsl:element>
               </xsl:when>
            <xsl:when test="preceding-sibling::idPart='1' and ancestor::Node[id='2']">
                   <xsl:element name="relatedValue">
                    <xsl:value-of select="substring-after(substring-before(.,')'),'
')"/>
                </xsl:element>
            </xsl:when>

            <xsl:when test="preceding-sibling::idPart='0' and ancestor::Node[id='3']">
```

```
                <xsl:element name="relatedValue">
                 <xsl:choose>
                     <xsl:when test="contains(substring(.,1,1),'-')">
                         <xsl:value-of select="substring-before(.,',')"/>
                     </xsl:when>
                     <xsl:otherwise>
                         <xsl:value-of select="concat('-',substring-before(.,','))"/>
                     </xsl:otherwise>
                 </xsl:choose>
             </xsl:element>
         </xsl:when>

         <xsl:when test="preceding-sibling::idPart='1' and ancestor::Node[id='3']">
                <xsl:element name="relatedValue">
                 <xsl:choose>
                     <xsl:when test="contains(substring(.,1,1),'-')">
                         <xsl:value-of select="substring-after(.,',')"/>
                     </xsl:when>
                     <xsl:otherwise>
                          <xsl:value-of select="concat('-',substring-after(.,','))"/>
                     </xsl:otherwise>
                 </xsl:choose>
             </xsl:element>
         </xsl:when>
       </xsl:choose>
   </xsl:template>
</xsl:stylesheet>
```

14.3. Příloha 3 – Diagram případů užití



Obrázek 23 - Příloha 3 - Diagram případů užití