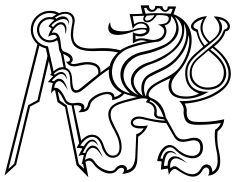
České vysoké učení technické v Praze
Fakulta stavební

Bakalářská práce

Praha 2012 Štěpán Turek

České vysoké učení technické v Praze
Fakulta stavební
Obor geoinformatika



Bakalářská práce

Implementace podpory WMS do programů GRASS GIS a SAGA GIS

Implementation of WMS support in GRASS GIS and SAGA GIS

Vedoucí práce: Ing. Martin Landa
Katedra mapování a kartografie

Praha 2012

Štěpán Turek

Abstract

Cílem této práce je zlepšit podporu WMS v programech GRASS GIS a SAGA GIS. Stávající podpora je v obou programech nedostatečná, což je nepříjemným omezením pro uživatele. Ti přicházejí o možnost využití řady zdrojů rastrových dat, která jsou prostřednictvím WMS serverů poskytována.

V úvodu práce je čtenář seznámen s principy fungování WMS z pohledu klienta, který komunikuje s WMS serverem.

Další části práce se věnují vytvořeným WMS modulům pro GRASS GIS a SAGA GIS. U obou modulů je popsán průběh jejich vývoje. Také jsou uvedeny návody pro práci s těmito moduly.

Klíčová slova: GIS, GRASS, SAGA, WMS

Abstract

The aim of the thesis is improvement of WMS support in GRASS GIS and SAGA GIS. Existing WMS support is inadequate in this software. This limits users who are not able to access many sources of raster data which are provided by WMS servers.

In the introduction, reader is got familiar with principles of WMS operation from the client's point of view.

Rest of the thesis deals with created WMS modules for GRASS GIS and SAGA GIS. There is described development of the modules and also there are included manuals for users.

Keywords: GIS, GRASS, SAGA, WMS

Prohlášení

Prohlašuji, že bakalářskou práci na téma „Implementace podpory WMS do programů GRASS GIS a SAGA GIS“ jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne ....

(podpis autora)

Poděkování

Chtěl bych zde poděkovat vedoucímu mé bakalářské práce, Ing. Martinu Landovi, za cenné rady a čas, který mi věnoval při konzultacích. Velké díky také patří mým rodičům za jejich podporu při studiu.

Obsah

Úvod

GRASS GIS (`http://grass.osgeo.org`) je geografický informační systém, šířený pod svobodnou licencí GNU GPL. Historie GRASS sahá do roku 1982. V tomto roce začala americká armáda s vývojem GRASS. Tento program ji měl pomoci se správou rozsáhlých oblastí, jež vlastní, aby byla schopna dostát požadavkům nové legislativy související s ochranou životního prostředí na majitele pozemků.

Významným milníkem pro GRASS byl rok 1995, kdy se americká armáda z tohoto projektu stáhla a kolem GRASS se začala rodit komunita dobrovolníků. Dnes je komunita vývojářů rozprostřena po celém světě, z nichž se většina rekrutuje z univerzit a výzkumných ústavů.

GRASS je jeden z nejrobustnějších svobodných GIS software. Umožňuje pracovat s vektorovými a rastrovými daty. Jelikož se GRASS vyvíjí již 25 let, jedním z dědictví takto dlouhého vývoje je, že jeho jádro je napsáno procedurálně v jazyce C. V poslední době je snaha vývojářů učinit GRASS použitelnější pro méně pokročilé uživatele. Z tohoto důvodu bylo vyvinuto nové GUI, které se intenzivně rozvíjí. Funkcionality GRASS jsou do programu implementovány v podobě modulů. Tyto moduly jsou do programu integrovány pomocí API, které existuje v C a Python verzi.

SAGA GIS (`http://www.saga-gis.org/`) je menší open source projekt, jehož vývoj začal na univerzitě v Goettingenu kolem roku 2004. Nyní je tento program šířen pod licencí GNU GPL. Vývojářská komunita je mezinárodní s těžištěm na domovské univerzitě. Hlavní síla SAGA GIS spočívá v práci s rastrovými daty. Program je schopen pracovat i s daty vektorovými. SAGA GIS je napsán v jazyce C++ a je rovněž koncipován modulárně. Moduly jsou do jádra integrovány pomocí API v jazyce C++.

Web Map Service (WMS) je standard[1], který definuje rozhraní mezi klientem a serverem pro získávání georeferencovaných dat v rastrových formátech [2] (např. JPEG, TIFF, PNG...). Jde o otevřený standard vyvíjený organizací OGC.[3].

[1] `http://www.opengeospatial.org/standards/wms`

[2] Vektorová data ve formátu SVG nebo CGM.

[3] `http://www.opengeospatial.org/`

1 Stručný úvod do WMS

Komunikace mezi klientem a serverem probíhá pomocí protokolu HTTP, kdy klient zašle serveru požadavek, server tento požadavek zpracuje a zašle klientovi soubor s odpovědí, což může být rastr ve formátu definovaném klientem, nebo soubor s metadaty.

WMS standard je definován v několika verzích:

Číslo verze	Rok vydání
1.0.0	2000
1.1.0	2001
1.1.1	2002
1.3.0	2004

Tabulka 1: Verze WMS

V dnešní době téměř všechny servery podporují verze 1.3.0 a 1.1.1., proto se dále text zaobírá pouze těmito verzemi.

1.1 Komunikace klient-server

Požadavek klienta na WMS server může být zaslán pomocí dotazovacích metod GET a POST protokolu HTTP. Pomocí těchto metod je klient schopen serveru předat parametry, na základě kterých server vytvoří odpověď. WMS standard vyžaduje podporu metody GET, zatímco podpora metody POST je volitelná.

1.1.1 HTTP GET

Metoda GET předává parametry jako součást URL. URL adresa je řetězec znaků reprezentující adresu zdroje informací. Tento řetězec má pevně danou strukturu:

```
protokol:// SERVER: port / cesta k~dokumentu ? parametry
```

Tomuto odpovídá tento příklad WMS dotazu metodou GET:

```
http://wms.cuzk.cz:80/wms.asp?REQUEST=GetCapabilities&VERSION=1.1.1
```

Protože port číslo 80 pro protokol HTTP je implicitní, není třeba jej zadávat. Aby server byl schopen rozpoznat jednotlivé části URL adresy, jsou určeny tyto znaky se speciálními funkcemi:

Znak	Funkce
?	Začátek řetězce parametrů.
&	Oddělovač parametrů.
=	Oddělovač názvu parametru a jeho hodnoty.
,	Oddělovač jednotlivých položek, pokud parametr obsahuje více hodnot
+	Reprezentuje mezeru.

Tabulka 2: Znaky v URL se speciální funkcí

Pokud je potřeba v URL uvést tyto vyhrazené znaky, lze použít URL kódování[4].

1.1.2 HTTP POST

Metoda POST neposílá parametry v URL adrese, ale přenáší je v těle POST zprávy.

1.2 Komunikace server-klient

Odpovědí serveru na WMS dotaz je soubor, který se odesílá protokolem MIME[5]. Tento protokol umožňuje zasílat soubory pomocí protokolu HTTP.

1.3 Fungování WMS v praxi

Nejčastěji uživatel získává data z WMS serveru pomocí klienta, který je součástí GIS nebo jiného programu. Každý klient na pozadí vytváří WMS dotazy a jejich tvorba je v této kapitole popsána na reálných příkladech.

[4] `hhttp://www.w3schools.com/tags/ref_urlencode.asp`
[5] `http://mgrand.home.mindspring.com/mime.html`

1.3.1 Dotaz typu GetCapapbilities

Jediné, co je potřeba znát pro zahájení komunikace s WMS serverem, je jeho URL. V tomto případě:

```
http://geoportal.cuzk.cz/WMS_ZABAGED_PUB/WMService.aspx
```

Nyní musí klient vytvořit dotaz typu GetCapabilities, aby zjistil informace o datech, která server poskytuje a o parametrech pro ostatní WMS dotazy (GetMap a GetFeatureInfo). Toto učiní přidáním parametrů k URL adrese WMS serveru:

```
http://geoportal.cuzk.cz/WMS_ZABAGED_PUB/WMService.aspx?
SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.3.0
```

Dotaz typu GetCapabilities obsahuje parametry, které jsou společné všem typům WMS dotazů, protože definují způsob komunikace.

- Parametr `SERVICE` sděluje serveru, že se jedná o WMS dotaz.
- Parametr `REQUEST` charakterizuje typ dotazu.
- Parametr `VERSION` popisuje, v jaké verzi WMS standardu je dotaz sestaven. Tento parametr může být vynechán pouze u GetCapabilities dotazu, u ostatních typů dotazů musí být vždy specifikován.

Aby byl server schopen správně zpracovat WMS dotazy, musí se s klientem dohodnout na verzi WMS, ve které bude probíhat následná komunikace. Toto je součástí dotazu typu GetCapabilities. Pokud není v dotazu GetCapabilities uveden parametr `VERSION`, server odpoví ve formátu nejvyšší podporované verze. Pokud klient explicitně požaduje určitou verzi, server odpoví v dané verzi, jestliže ji podporuje. Jak bylo výše zmíněno, dnes naprostá většina serverů podporuje verze WMS standardu 1.1.1 a 1.3.0. Pokud klient podporuje tyto dvě verze, problém s nekompatibilitou v podstatě odpadá.

Na tento dotaz server vrátí soubor ve formátu XML.

XML je značkovací jazyk, který se používá pro výměnu dat mezi aplikacemi. Základním prvkem XML je element, který je vymezen počáteční a ukončovací značkou.

Každý element je vnořen do jiného elementu, s výjimkou kořenového elementu. Kořenový element obsahuje všechny ostatní elementy XML dokumentu. Elementy se nesmí křížit, což znamená, že počáteční a ukončující značka musí být vnořena stejnému elementu.

Element může mít obsah, což je text mezi počáteční a ukončovací značkou a také obsahovat atributy. Atribut lze popsat jako proměnnou s textovou hodnotou.

```
 <Layer queryable="1">
        <Name>Default</Name>
        <Title>Default</Title>
</Layer>
```

V tomto příkladě je kořenovým elementem `Layer`, který má definován atribut `queryable` s hodnotou 1. Tento element má dva vnořené elementy `Name` a `Title`. Oba tyto elementy mají stejný obsah, a sice `Default`.

Informace o verzi WMS standardu XML souboru je uvedena jako atribut kořenového elementu:

```
<WMS_Capabilities...    ...version="1.3.0">
```

Přímými potomky kořenového elementu jsou elementy `<Service>` a `<Capability>`.

Element `<Service>` obsahuje informace o WMS Serveru a poskytovateli dat. Druhý element `<Capability>` je mnohem důležitější, protože poskytuje všechny informace, které jsou potřeba pro další komunikaci s WMS serverem.

```
<Capability>
    <Request>
          ...
       <GetMap>
            <Format>image/png</Format>
            <Format>image/jpeg</Format>
          ...
       </GetMap>
       <GetFeatureInfo>
            <Format>text/html</Format>
            <Format>text/xml</Format>
              ...
       </GetFeatureInfo>
           ....
```

V této části jsou uvedeny formáty odpovědí ve tvaru protokolu MIME pro dotaz typu GetMap a GetFeatureInfo. V tomto případě WMS server poskytuje mapy jako rastry ve formátu PNG, JPEG a na dotaz typu GetFeatureInfo může vrátit odpověď ve formátu HTML nebo XML.

Velmi důležitý je element `<Layer>`, který obsahuje informace o mapové vrstvě. Všechny vrstvy jsou uspořádány do stromové struktury s jedním kořenovým elementem.

```
<Capability>
    ...
  <Layer>
   <Title>ZABAGED</Title>
     <CRS>EPSG:3035</CRS>
     <CRS>EPSG:3034</CRS>
     <CRS>EPSG:4326</CRS>
      ...
     <BoundingBox CRS="EPSG:3035" minx="4434628.0972282" miny="2778319.58676976"
                                  maxx="4987359.29769667" maxy="3190250.19895492"/>
     <BoundingBox CRS="EPSG:3034" minx="4109720.95957183" miny="2382975.60863023"
                                  maxx="4643932.77142764" maxy="2780500.92255097"/>
      ...
```

Zde je uveden název kořenové vrstvy `<Title>` a projekce `<CRS>`, v nichž je dostupná. Pokud se jedná o verzi WMS 1.1.1., projekce je definována pod názvem`SRS`.

Název vrstvy se uvádí pomocí elementů `<Title>` a `<Name>`. Element `<Title>` je název vrstvy ve formátu pro člověka pochopitelném a má pouze informativní charakter, zatímco element `<Name>` slouží jako unikátní klíč, pod kterým je možné danou vrstvu jednoznačně identifikovat v rámci WMS serveru. Jelikož kořenová vrstva nemá element `<Name>`, není možné poslat požadavek na data této vrstvy. Vrstvy, které neposkytují žádná data, se uvádí z důvodu dědičnosti.

Jelikož projekce je atribut, který je v rámci stromu vrstev děděn, a v příkladu jsou uvedeny atributy kořenového elementu `<Layer>`, jsou všechny vrstvy tohoto serveru dostupné v těchto projekcích. Jakákoliv vrstva může mít definovány další projekce, které budou děděny všemi jejími následovníky ve stromu.

Element `<BoundingBox>` reprezentuje obdélník definovaný minimálními a maximálními souřadnicemi v systému projekce uvedené v atributu `CRS`, který vymezuje

rozsah poskytovaných dat. Tento element se také ve stromu dědí, pokud je však v potomcích vrstvy nově definován pro stejnou projekci, nahrazuje děděný element.

Právě dědičnost je důvodem, proč jsou vrstvy uspořádány do stromové struktury. Díky této struktuře není potřeba v tomto případě u každé vrstvy uvádět všech 16 souřadnicových systémů a obdélníků, čímž dochází k úspoře dat, která jsou přenášena mezi serverem a klientem a také k větší přehlednosti a stručnosti XML souboru.

```
<Capability>
  <Layer>
    <Title>Kořenová vrstva bez dat, chybí name</Title>
    <Layer>
      <Layer>
        <Title>Vrstva č. 1</Title>
        <Name>vrstva1</Name>
      </Layer>
    <Layer>
    <Layer>
      <Layer>
        <Title>Vrstva č. 2</Title>
        <Name>vrstva2</Name>
        <Layer>
          <Title>Vrstva č. 3</Title>
          <Name>vrstva3</Name>
        </Layer>
        <Layer>
          <Title>Vrstva č. 4</Title>
          <Name>vrstva4</Name>
        </Layer>
      </Layer>
    </Layer>
</Capability>
```

Tato delší ukázka ilustruje uspořádání vrstev ve stromové struktuře. Kořenová vrstva se větví v první úrovni na vrstvy `vrstva1`, `vrstva2`. Vrstvy `vrstva1`, `vrstva3` a `vrstva4` jsou tzv. listy stromu. Takto se nazývají elementy ve stromové struktuře, které nemají žádné potomky.

```
<Layer queryable="0" opaque="false" noSubsets="0">
    <Name>GR_CR4</Name>
    <Title>MČR 1 : 1 000 000</Title>
    <Style>
        <Name>Default</Name>
        <Title>Default</Title>
          ...
    </Style>
</Layer>
```

Každá vrstva reprezentuje určitá data, která mohou být zobrazena rozličnými způsoby. Způsob zobrazení definuje element `<Style>`. V tomto případě je pro vrstvu `MČR 1 : 1 000 000` dostupný pouze jeden styl. Vztah elementů `<Name>` a `<Title>` v elementu `<Style>` je stejný jako v elementu `<Layer>`.

1.3.2 Dotaz typu GetMap

Díky dotazu GetCapabilities získá klient všechny potřebné informace ke stažení dat z WMS serveru. Toto provede pomocí dotazu typu GetMap v tomto tvaru:

```
http://geoportal.cuzk.cz/WMS_ZABAGED_PUB/WMService.aspx?
SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&
LAYERS=GR_CR4&STYLES=Default&FORMAT=image/png&CRS=EPSG:4326&
BBOX=48.093144621684,11.6163532829661,51.4980192528993,19.0628256634265&
WIDTH=800&HEIGHT=600&TRANSPARENT=true
```

- Povinný parametr `FORMAT` definuje formát, v němž bude vygenerován rastr odpověďi.
- Povinný parametr `LAYERS` obsahuje vrstvy, které mají být zobrazeny ve výsledném rastru. Jednotlivé vrstvy se oddělují čárkou a jsou vykresleny podle pořadí, v jakém jsou uvedeny. Vrstva překrývá ty vrstvy, jenž jsou ve výčtu od ní vpravo a je překryta vrstvami, které se nacházejí vlevo.
- Povinný parametru `STYLES` obsahuje styly vybraných vrstev. Styly se uvádí ve stejném pořadí jako vrstvy a také jsou odděleny čárkou.
- Povinný parametr `CRS` definuje projekci výsledné mapy. Ve verzi 1.1.1. se parametr nazývá `SRS`.

- Povinný parametr `BBOX` určuje v jednotkách požadované projekce obdélník, ze kterého jsou data požadovány. Hodnoty mohou být i vně obdélníku definovaném v GetCapabilites, protože tento obdélník pouze informuje o rozsahu poskytovaných dat.
- Povinné parametry `WIDTH` a `HEIGHT` definuje počet pixelů získaného rastru.
- Parametr `TRANSPARENT` udává, zda plochy, které nereprezentují žádná data, budou průhledné (hodnota true) nebo budou mít barvu definovanou v parametru `BGCOLOR` (hodnota false). Barvy se uvádí v hexadecimálním RGB formátu. Jestliže parametr `BGCOLOR` není specifikován, je implicitně nastavena bílá barva.

Všechny tyto parametry byly zvoleny na základě informací, které byly získány pomocí dotazu GetCapabilities.

Odpovědí na tento dotaz je rastr ve formátu PNG s přehledovou mapou České republiky:

1.3.3 Dotaz typu GetFeatureInfo

Poslední typ dotazu je volitelný a nazývá se GetFeatureInfo. Tento dotaz, na rozdíl od předchozích, nemusí server podporovat. Dotaz slouží k získání informací o prvcích ve vrstvě. Pokud je v elementu `<Layer>` uveden atribut `queryable` s hodnotou `1`, je možné na tuto vrstvu aplikovat dotaz GetFeatureInfo.

Jako příklad pro tento typ dotazu je použit WMS server Středočeského kraje. Jedna z WMS služeb, které tento server poskytuje, jsou zóny integrovaného dopravního systému dostupné na :`http://mapy.kr-stredocesky.cz/ids_zony_wms`.

Nejprve se dotazem GetaCapabilities zjistí informace o WMS serveru a jím poskytovaných datech.

```
http://mapy.kr-stredocesky.cz/ids_zony_wms?
SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.3.0
```

Z odpovědi je patrné, že server obsahuje pouze jednu vrstvu `Zony SID`, se kterou lze dále pracovat, protože její rodičovská vrstva nemá atribut `<Name>` nutný pro dotazy GetMap a GetFeatureInfo.

Vrstva je dostupná v jediném souřadnicovém systému. Za povšimnutí stojí opakované uvedení projekce a stejného obdélníku `BoundingBox`, v němž jsou poskytována data jak v kmenové vrstvě, tak v jejím potomku `Zony SID`. Jelikož se tyto elementy vrstvy dědí, stačilo by je uvést pouze v kořenové vrstvě.

Na základě informací z dotazu GetCapabilities byl vytvořen dotaz GetMap:

```
http://mapy.kr-stredocesky.cz/ids_zony_wms?
SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&
&LAYERS=sid_zony&FORMAT=image/png&CRS=EPSG:2065&
BBOX=48.093144621684,11.6163532829661,51.4980192528993,19.0628256634265&
WIDTH=1000&HEIGHT=1000
```

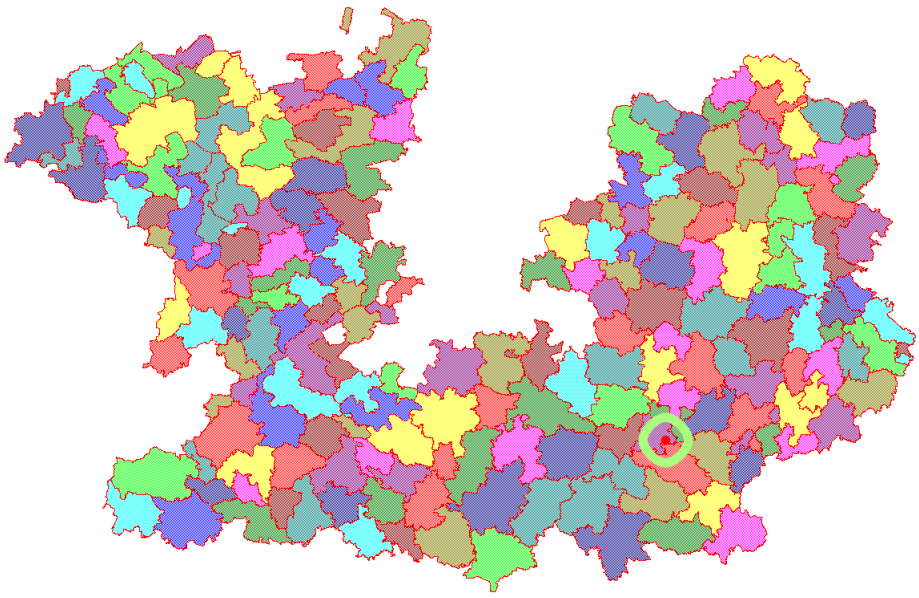
A výsledný dotaz GetFeatureInfo vypadá takto:

```
http://mapy.kr-stredocesky.cz/ids_zony_wms?
REQUEST=GetFeatureInfo&VERSION=1.3.0&
FORMAT=image/png&LAYERS=sid_zony&CRS=EPSG:2065&
BBOX=-834258.9702,-1129697.611,-651189.0329,-968639.3882&WIDTH=1000&HEIGHT=1000&
INFO_FORMAT=text/html&I=695&J=720&QUERY_LAYERS=sid_zony
```

Jak lze vidět, tento dotaz zahrnuje parametry předchozího dotazu GetMap a přidává další:

- Parametr `REQUEST` uvádí typ dotazu GetFeatureInfo.
- Parametr `INFO_FORMAT` popisuje v MIME tvaru formát odpovědi na tento dotaz. WMS standard neuvádí implicitní formát, který musí server podporovat. V tomto případě se jedná o HTML soubor.
- Parametry `I` a `J` lokalizují prvek, na který dotaz GetFeatureInfo směřuje. Tyto souřadnice reprezentují souřadnicový systém obrázku s jednotkami v pixelech a začátkem v levém horním rohu. Souřadnice os `I` a `J` narůstají směrem vpravo resp. dolů. Interval souřadnic je dán parametry `WIDTH` a `HEIGHT`, kdy `I`, `J` mají rozsah od 0 do `WIDTH` - 1 resp. od 0 do `HEIGHT` - 1. Ve verzi WMS 1.1.1. se parametry `I` a `J` označují jako `X` resp. `Y`.
- `QUERY_LAYERS` je výčet vrstev, kterých se týká dotaz GetFeatureInfo.

Výsledek GetMap dotazu s bodem o souřadnicích `I`, `J` `695` a `720`:



A jako odpověď server zašle HTML dokument, který se v prohlížeči zobrazí takto:

Pokud je serveru položen WMS dotaz ve špatném tvaru, vrátí výjimku implicitně ve formátu XML souboru, v němž je blíže specifikována chyba v dotazu.

2 WMS modul pro GRASS

Jelikož původně GRASS neměl žádné GUI, jsou jeho moduly přizpůsobeny pro práci v příkazové řádce. Každý modul si lze představit jako funkci, která má definované vstupní argumenty a určené výstupy. Nevýhodou tohoto přístupu je, že modul není schopen s uživatelem komunikovat za běhu. Uživatel může jeho chování ovlivnit jen pomocí argumentů, které jsou zadány před spuštěním.

Z tohoto důvodu implementace modulu odpovídá dotazu typu GetMap, kdy musí být zadány parametry tohoto dotazu a modul se postará o stažení rastru, jeho import do GRASS a případné další úpravy.

Tento charakter modulu nebrání úplné interaktivní implementaci podpory WMS do GRASS. Je ji ovšem potřeba rozdělit do dvou částí. První částí je vytvoření modulu, který umožní pracovat s WMS daty i uživatelům, jenž používají pouze příkazovou řádku. Do GUI programu je pak možné implementovat interaktivní část.

2.1 Analýza původního stavu

V GRASS GIS již existuje modul *r.in.wms*[6] umožňující získání WMS dat. Velkým problémem tohoto modulu je rozdělení jeho kódu do 8 souborů, což jej činí velmi nepřehledným. Tento modul byl původně napsán jako shell skript. V aktuální vývojářské verzi GRASS 7 byly všechny moduly v shell skriptu přepsány do jazyka Python, což také nepřispělo k jeho zpřehlednění.

Ještě větším problémem však je, že tento modul selhává při komunikaci s mnoha WMS servery. Z vlastní zkušenosti musím konstatovat, že při používání modul spíše nefungoval než pracoval správně.

Na základě těchto faktů bylo rozhodnuto, že se WMS modul pro GRASS vytvoří od základů znova. Případná oprava chyb ve stávajícím modulu a reorganizace kódu by byla mnohem problematičtější a časově náročnější.

[6] `http://grass.fbk.eu/grass64/manuals/html64_user/r.in.wms.html`

2.2 Stanovení cílů implementace

Návrh struktury modulu byl realizován na základě těchto požadavků:

- Modul bude plně podporovat všechny možnosti dotazu GetMap.
- Modul bude komunikovat se serverem prostřednictvím knihovny GDAL[7] a také pomocí vlastní implementace.
- WMS severy mají nastaveny limity pro přenos dat, aby zabránily dotazům, které by je nadměrně zatížily. Tyto limity jsou dány formou maximálních hodnot parametrů `WIDTH` a `HEIGHT` dotazu GetMap. Modul bude umět požadavek rozdělit na více WMS dotazů a získat tak požadovaný rastr po částech tzv. dlaždicích, které posléze složí do jednoho rastru.
- Důležitým prvkem při práci v systému GRASS je lokace. Lokace sdružuje data, která mají stejnou projekci. Její podmnožinou je tzv. mapset seskupující data lokace do logických celků. Na začátku práce v GRASS uživatel vybere lokaci, ve které chce pracovat. Následná práce je svázána s touto lokací a její projekcí.

  Pokud by uživatel chtěl získat rastr z WMS serveru, který neposkytuje data v projekci lokace, musel by nejprve vytvořit lokaci v projekci WMS dotazu a poté tyto data manuálně transformovat a zkopírovat do pracovní lokace.

  Proto bude modul schopen obdržená data automaticky transformovat do souřadnicového systému lokace, pokud se jejich projekce bude lišit.
- Jelikož existují další rozšíření standardu WMS, jako například WMTS [8], struktura modulu umožní snadnou implementaci těchto rozšíření do existujícího kódu.
- Vstupní argumenty modulu budou kompatibilní s modulem *r.in.wms*. Některé nevýznamné argumenty, které nemají vliv na funkčnost modulu, mohou být vynechány.
- Doplňkovou funkcí modulu bude možnost stáhnout a vypsat na standardní výstup obsah Capabilities souboru. Další zpracování tohoto výstupu bude součástí GUI.

[7] `www.gdal.org/`

[8] `http://www.opengeospatial.org/standards/wmts`

2.3 Volba způsobu implementace

V GRASS GIS verze 7 mohou být moduly implementovány v jazyce C nebo Python. V jazyce C se implementují moduly, které jsou náročné na výpočetní výkon počítače. Rychlost je vykoupena značně delším vývojovým cyklem, neboť programátor je nucen se starat o mnoho věcí, o které se Python postará sám.

Koncept WMS modulu byl zvolen tak, aby neprováděl žádné složité výpočetní operace, ale aby pro tyto typy operací byly využity již implementované funkcionality ostatních GRASS modulů nebo knihoven. Proto byl vybrán jazyk Python.

Pro práci modulu se staženým rastrem jako je transformace a spojení dlaždic byla zvolena knihovna GDAL, která je součástí standardní instalace GRASS.

Tato open source knihovna umožňuje čtení, zápis a reprojekci rastrů. Základním prvkem knihovny, který reprezentuje rastrová data je dataset. Každý dataset reprezentuje rastrová data v určitém formátu, s nimiž pracuje pomocí ovladače. Ovladač je třída, která je schopná číst a zapisovat data v určitém formátu.

2.4 Volba vstupních argumentů modulu

Vstupní argumenty budou reprezentovat jednotlivé parametry dotazu GetMap. Počet řádků (`HEIGHT`), sloupců (`WIDTH`) a geografický rozsah dat (`BBOX`) budou reprezentovány pomocí regionu.

Region v GRASS je datová struktura, která definuje oblast na základě obdélníku. Tento obdélník je dán mezními kartografickými souřadnicemi v každé ose a počtem řádků a sloupců. Každý region je vztažen k určité projekci, která je totožná s projekcí lokace, ve které je uložen.

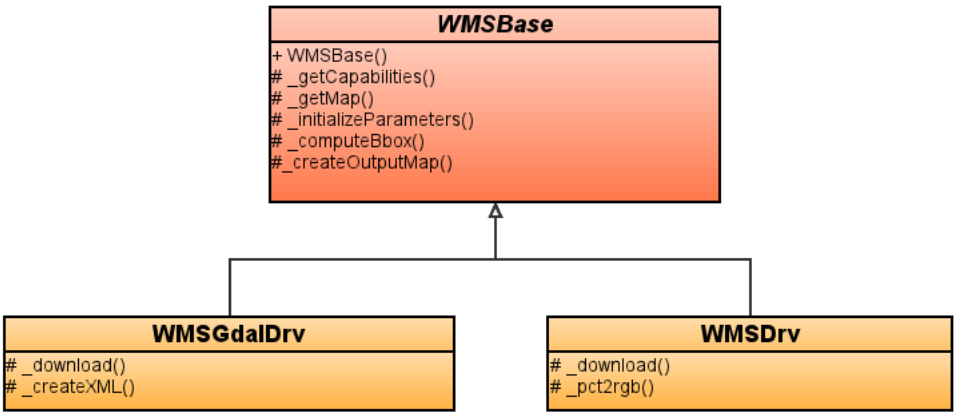
Regionu určuje rozsah, na kterém bude aplikována činnost modulů. Výjimkou jsou moduly, které data načítají, jako je například modul *r.in.gdal*[9]. Tyto moduly implicitně načítají data v celém jejich rozsahu, bez ohledu na výpočetní region.

Vstupní argumenty, které jsou nad rámec parametrů dotazu GetMap, se týkají dlaždicování. Modul umožňuje zadání maximálního počtu řádků a sloupců v jednom WMS dotazu. Na základě těchto hodnot se rozdělí stažení dat z WMS serveru do několika dotazů a výsledný rastr je složen z rastrů získaných těmito dotazy.

[9] `http://grass.fbk.eu/grass64/manuals/html64_user/r.in.gdal.html`

2.5 Implementace modulu

Každý GRASS modul napsaný v jazyce Python obsahuje funkci `main`. Tato funkce je volána při spuštění modulu. U WMS modulu funkce `main` pouze vytvoří instanci tříd `WMSGDALDrv` nebo `WMSDrv` v závislosti na volbě uživatele a vše ostatní se odehrává v rámci těchto instancí. Struktura modulu je popsána v následujícím UML diagramu zahrnujícím nejdůležitější třídy a jejich významné metody.



Třída `WMSBase` je abstraktní třída, která vykonává ty funkce, které se netýkají komunikace s WMS serverem, jež je implementována v odvozených třídách `WMSGDALDrv` a `WMSDrv`.

2.5.1 Třída `WMSBase`

Hlavními funkcemi třídy `WMSBase` je výpočet parametrů pro komunikaci s WMS serverem, import již stažených dat do lokace a dodatečné úpravy těchto dat pomocí modulů GRASS.

Tato třída obsahuje tyto důležité metody:

- `_GetCapabilities` - Vytvoří a pošle WMS serveru dotaz typu GetCapabilities, následně obdrženou odpověď vypíše na standardní výstup a tímto se běh modulu ukončí.
- `_GetMap` - Tato metoda postupně volá níže zmíněné metody v pořadí v jakém jsou vedeny.

- `_initializeParameters` - Metoda inicializuje proměnné potřebné pro další běh modulu. Jde o uložení hodnot ze slovníků vstupních argumentů do separátních proměnných a také získání hodnot ze zvoleného regionu.

- `_computeBbox` - Pokud se liší projekce, ve které budou data stažena z WMS serveru a projekce pracovní lokace, je potřeba souřadnice, které vymezují region, transformovat do projekce WMS dotazu, aby bylo možné definovat parametr `BBOX` ve správném souřadnicovém systému.

  Zpravidla výsledkem transformace není obdélník se stranami rovnoběžnými se souřadnicovými osami systému, ale obecný čtyřúhelník. Protože parametr `BBOX` WMS dotazu GetMap musí být uveden v tomto obdélníku, jsou vybrány extrémní souřadnice, které tvoří obdélník rovnoběžný s osami souřadnicového systému.

- `_download` - Jedná se o metodu virtuální, definovanou v potomcích třídy, která dotazem GetMap stáhne rastr a uloží jej do dočasného souboru. Zpravidla jsou rastrová data poskytována WMS serverem tříkanálová reprezentující RGB systém barev. Pokud jde o rastr s průhlednými plochami, obsahuje ještě alfa kanál, který definuje průhlednost pixelů. Návratovou hodnotou metody je cesta k souboru s rastrem.

- `_createOutputMap` - Pokud je to potřeba, tato metoda nejprve pomocí utility *gdalwarp*[10] knihovny GDAL transformuje rastr do projekce lokace. Transformovaný rastr je uložen do nového souboru. Jelikož rastr je matice o určitých počtech sloupců a řádků, čemuž ale tvar transformovaného rastru zpravidla již neodpovídá, je potřeba mít informaci o tom, který pixel odpovídá původnímu rastru, a který pixel již nereprezentuje data původního rastru. Tato informace je zahrnuta do alfa kanálu. Pixely, které do původního rastru nepatří, mají hodnotu v alfa kanálu plně průhledného pixelu. Pokud transformovaný rastr neobsahuje alfa kanál, utilita *gdalwarp* jej vytvoří.

Posléze pomocí modulu *r.in.gdal* importuje rastrová data do pracovní lokace. Tento modul vytvoří v pracovní lokaci rastrové vrstvy odpovídající jednotlivým

[10] `http://www.gdal.org/gdalwarp.html`

kanálům importovaného rastru. Tyto vrstvy jsou pomocí modulu *r.composite*[11] sloučeny do jedné barevné vrstvy.

Aby byly správně interpretovány průhledné plochy výsledné rastrové vrstvy, vytvoří se před spuštěním modulu *r.composite* z alfa kanálu inverzní maska. Maska se v GRASS používá v těch případech, kdy rozsah dat, nad nimiž bude modul pracovat, nelze vyjádřit pomocí regionu. Maska je v GRASS reprezentována rastrovou vrstvou, jejíž název je `MASK`. Pixely, kde je maska definována nejsou brány při výpočtu v úvahu.

Při použití masky vytvoří modul *r.composite* barevný rastr i s průhlednými pixely definovanými v alfa kanálu.

WMS modul za svého běhu vytvoří několik dočasných souborů. Tyto dočasné soubory jsou smazány ihned poté, co již nejsou potřeba. Pokud však uživatel neočekávaně přeruší běh programu, může se stát, že některý soubor nebude smazán. Aby se tomuto zabránilo, je v destruktoru třídy, který se volá i při neočekávaném ukončení modulu, provedeno smazání těchto souborů, jenž nebyly odstraněny. Toto je provedeno i s vrstvami, které reprezentují jednotlivé kanály a maskou.

2.5.2 Třída `WMSGDALDrv`

Tato třída přistupuje k datům WMS serveru prostřednictvím WMS ovladače knihovny GDAL. Parametry WMS dotazu jsou ovladači předány ve formě XML souboru vytvořeném metodou `_createXML`. [12]. Metotda `_download` následně stáhne data prostřednictvím GDAL ovladače a uloží je do souboru.

2.5.3 Třída `WMSDrv`

Tato třída komunikuje se serverem přímo, bez použití další knihovny. Nejprve je ze vstupních parametrů vytvořen WMS dotaz typu GetMap. Pak následuje výpočet rozměrů dlaždic v souřadnicovém systému projekce, rozměru dlaždic v pixelech a jejich počet. Na základě těchto hodnot jsou v cyklu staženy jednotlivé dlaždice, kdy je k WMS dotazu přidán parametr `BBOX` pro konkrétní dlaždici, tento dotaz je poslán

[11] `http://grass.fbk.eu/gdp/html_grass64/r.composite.html`
[12] `http://www.gdal.org/frmt_wms.html`

WMS serveru a do dočasného souboru je uložen rastr z odpovědi na dotaz. Spojování dlaždic do jednoho rastru je řešeno pomocí knihovny GDAL. Při prvním průchodu cyklu se vytvoří nový dataset, kde se dlaždice postupně, tak jak jsou stahovány, spojují.

2.6 Zajímavé problémy a jejich řešení

Při implementaci třídy `WMSDrv` se vyskytl problém při dlaždicování. Při stažení barevného jednokanálového rastru (např. PNG, GIF) s přiloženou globální tabulkou barev vznikl po spojení jednotlivých dlaždic rastr, který tuto tabulku neobsahoval.

Protože v tomto případě konkrétní barvy jednotlivých kanálů jsou definovány až v tabulce barev, není možné bez této tabulky správně interpretovat hodnoty pixelů, které zde představují pouze odkazy na položky tabulky. V různých dlaždicích jsou stejné barvy definovány jinými položkami tabulky barev a hodnota pixelů na ně odkazující se v nich liší. Při spojování dlaždic knihovna GDAL nebrala tabulku barev v úvahu a interpretovala přímo hodnoty těchto pixelů. Výsledný rastr se spojenými dlaždicemi vypadal například takto:

Tento problém byl vyřešen pomocí upraveného kódu GDAL utility *pct2rgb* [13]. Tento kód obsahuje metoda `_pct2rgb` třídy `WMSGDALDrv`. Tato metoda vytvoří 4

[13] `www.gdal.org/pct2rgb.html`

kanálový dataset (RGB + alfa vrstva) a do jednotlivých kanálů jsou pixel po pixelu přiřazovány jejich hodnoty z tabulky barev.

2.7 Ukázka práce modulu

Nyní je modul dostupný pod názvem *r.in.wms2* v repozitáři *GRASS AddOns* [14]. V tomto repozitáři jsou uloženy nové funkcionality implementované do GRASS, které je ještě potřeba otestovat uživateli a dokončit jejich vývoj. Až potom lze tyto funkcionality začlenit přímo do programu.

Modul se z repozitáře do GRASS nainstaluje pomocí tohoto příkazu:

```
g.extension extension=r.in.wms2
```

2.7.1 Vstupní argumenty

V GRASS GIS mají moduly dva typy vstupních argumentů. Argumenty, které nesou binární informaci 0/1, jsou modulu předány ve formě voleb, což řetězce uvedené za pomlčkou. Pokud je volba uvedena mezi argumenty při spuštění modulu, nese hodnotu 1, pokud uvedena není, je ji přiřazena hodnota 0.

Volba	Popis
`-o`	Data nebudou obsahovat průhlednou vrstvu.
`-c`	Vypíše na výstup Capabilities soubor.
`-d`	Nepoužije knihovnu GDAL pro komunikaci s WMS serverem, ale vlastní řešení.

Tabulka 3: Volby modulu r.in.wms2

Dalším typem vstupních argumentů je parametr. Parametr obsahuje jako hodnoty textové řetězce. I v případě, že argument obsahuje číselnou hodnotu, je tato hodnota uložena jako text a v modulu musí být převedena na číselný typ.

Volby a parametry jsou modulu předány ve slovnících `flags` resp. `options`. Tyto slovníky v položkách obsahují název volby resp. parametru a jejich hodnotu.

[14] `http://svn.osgeo.org/grass/grass-addons/`

Vstupní parametry modulu jsou:

- `output` - Povinný parametr. Název vrstvy vytvořené modulem v lokaci.
- `mapserver` - Povinný parametr. URL adresa WMS serveru.
- `layers` - Povinný parametr. Uvádí se ve stejném formátu jako v getMap parametru `LAYERS`.
- `srs` - Číslo `EPSG` kódu, který definuje projekci WMS dotazu.
- `region` - Název regionu. Pokud není uveden, je použit aktuálně nastavený region.
- `wms_version` - Verze WMS standardu, lze uvést `1.1.1` (implicitní), nebo `1.3.0`
- `format` - Formát rastru WMS dotazu. Možné hodnoty jsou `geotiff` (implicitní), `tiff`, `jpeg`, `gif`, `png`.
- `method` - Pokud projekce rastru z WMS dotazu a projekce lokace nejsou stejné, tento parametr definuje jakou metodou bude rastr transformován do projekce lokace. Může mít tyto hodnoty: `near` (implicitní), `bilinear`, `cubic`, `cubicspline`.
- `maxrows` - Číslo, které definuje maximální počet pixelů v řádce jedné dlaždice. Implicitně `400`.
- `maxcols` - Číslo, které definuje maximální počet pixelů ve sloupci jedné dlaždice. Implicitně `300`.
- `urlparams` - Zde lze uvést další parametry WMS dotazu GetMap. Pouze s volbou `-o`, protože GDAL WMS ovladač toto neumožňuje.
- `styles` - Styly vrstev ve stejném formátu jako v getMap parametru `STYLES`.
- `bgcolor` - Pokud je při spuštění modulu uvedena volba -o, budou všechny plochy, které by byly průhledné, vyjádřeny barvou uvedenou v tomto parametru. Barva se uvádí v hexadecimálním RGB formátu. Implicitní je bílá barva (`0xFFFFFF`).

2.7.2 Příklad

Jako příklad je použit WMS server[15] poskytující družicová data. Na základě informací získaných pomocí GetCapabilities dotazu, je možné spustit modul například s těmito parametry:

```
r.in.wms2 -d mapserver=http://iceds.ge.ucl.ac.uk/cgi-bin/icedswms
layers=bluemarble,landsat_1_01 styles=default,default
output=landsat srs=4326 format=png maxcols=500 maxrows=600
```

Při spuštění modulu byla projekce lokace totožná s projekci rastru z WMS dotazu (EPSG:4326). Region byl dán obdélníkem o geografických souřadnicích -20° z. d., 30° s. š., 40° v. d., 65° s. š. a počet pixelů v řádcích byl 700 a ve sloupcích 1200. Protože byl počet pixelů v obou dimenzích větší než parametry `maxcols` a `maxrows`, muselo být stažení rastru rozděleno na stažení 4 dlaždic, které byly následně spojeny. Data z WMS serveru vlastní implementací, protože byla uvedena volba `-d`.

Pro práci s regiony je určen modul *g.region*[16]. Pomocí tohoto modulu je možné modifikovat parametry regionu, vytvářet regiony nové nebo zjišťovat hodnoty parametrů regionu.

Takto vypadá stažený rastr:

[15] http://iceds.ge.ucl.ac.uk/cgi-bin/icedswms
[16] http://grass.fbk.eu/gdp/html_grass64/g.region.html

Modul *r.in.wms* v tomto případě, jako v mnoha ostatních, neuspěje. Jelikož vstupní argumenty modulu *r.in.wms2* jsou kompatibilní s argumenty modulu *r.in.wms*, stačí pouze změnit název modulu v příkazu na *r.in.wms* a spustit jej se stejnými argumenty:

```
r.in.wms -d mapserver=http://iceds.ge.ucl.ac.uk/cgi-bin/icedswms
layers=bluemarble,landsat_1_01 styles=default,default output=landsat srs=432
format=png maxcols=500 maxrows=600
```

2.8 Další vývoj

Po otestování uživateli by se měl modul *r.in.wms2* stát součástí GRASS a nahradit modul *r.in.wms*. Pokud k tomu dojde, bude pravděpodobně přejmenován na *r.in.wms*.

Tento modul je pouze jednou částí integrace WMS do GRASS. Nyní bude dalším krokem vytvoření interaktivního WMS řešení, které by se uživatelskou přívětivostí přiblížilo programům ArcGIS[17] nebo QGIS[18]. Tyto programy mají z uživatelského pohledu velmi dobrou podporu WMS.

Idea tohoto řešení je, že uživatel si na základě dialogu vygenerovaného z Capabilities souboru vybere data, která chce zobrazit a v GRASS GUI se přidá nová WMS vrstva. Tato vrstva bude dynamicky získávat data, v závislosti na aktuálním rozsahu mapového okna.

Ve vývojové verzi GRASS 7 již existuje dialogové okno, které na základě vložené URL adresy stáhne Capabilities soubor a z tohoto souboru zobrazí dostupné vrstvy.

Tento dialog funguje tak, že po zadání URL adresy se zavolá modul *r.in.wms* a ten na standardní výstup vypíše seznam vrstev z Capabilities souboru. Tyto vrstvy jsou pak zobrazeny v dialogovém okně.

Aby tento dialog umožnil uživateli využít všech možností WMS serveru, nestačí jen vypsat seznam vrstev, ale je také potřeba zobrazit k výběru dostupné formáty, projekce uživatelem vybraných vrstev, styly a také metadata s informacemi o vrstvách. K tomu nestačí jen prostý výpis seznamu vrstev, ale je nutné pracovat s celým obsahem Capabilities souboru.

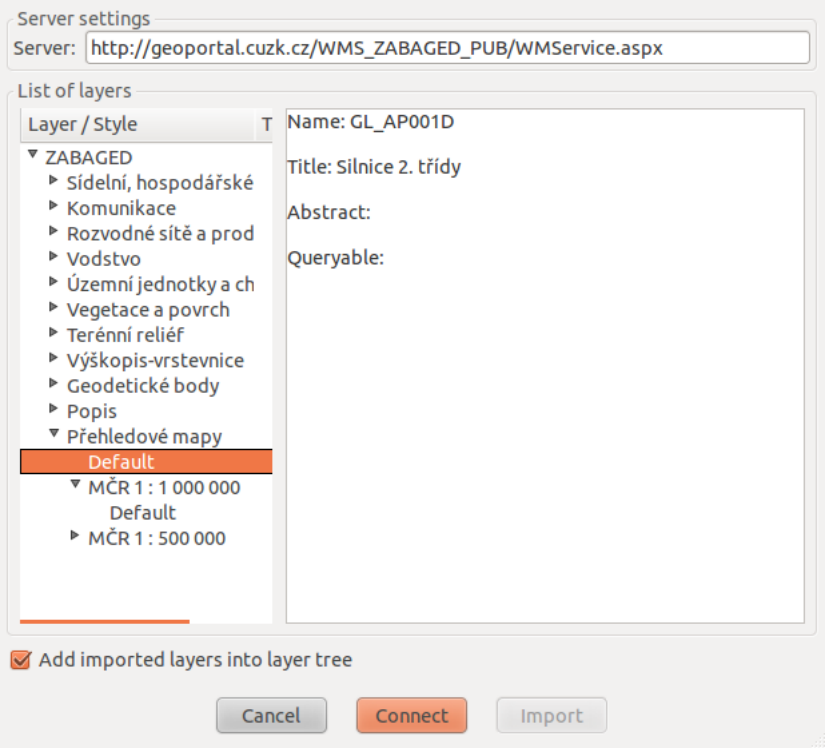
[17] `http://www.arcgis.com/`
[18] `http://www.qgis.org/`

V rámci bakalářské práce byly již učiněny některé kroky k vylepšení tohoto dialogu.

Dialog nyní využívá modul *r.in.wms2*, který umožňuje vypsat celý obsah Capabilities souboru na standardní výstup. Tento výstup je načten a posléze zpracován do takové datové struktury, která ke každé vrstvě umožňuje zjistit seznam elementů včetně těch děděných, což je největší problém při zpracování Capabilities souboru. Na základě této struktury lze v dialogu zobrazit strom vrstev, který odráží uspořádání elementů `<Layer>` v Capabilities souboru.

Dialog obsahuje textové pole, které bude sloužit k informací o jednotlivých vrstvách. V aktuální podobě vypisuje pouze část informací.

Takto vypadá aktuální verze WMS dialogu v GRASS:



Dalším krokem bude v GUI vytvořit rastrovou WMS vrstvu, která, pokud bude zobrazena v mapovém okně, bude zobrazovat WMS data pomocí modulu *r.in.wms*. Tento modul bude spouštěn s parametry, které budou zadány v dialogovém okně před přidáním vrstvy. Při každém volání z mapového okna se změní pouze parametr region, který bude odpovídat aktuálnímu rozsahu a rozlišení pixelu mapového okna.

Data rastru se v mapovém okně vykreslují na základě dvou parametrů. Těmito parametry jsou rozsah, který je určen souřadnicemi projekce a rozměry pixelu. Pokud se změní rozsah i rozměr pixelu dochází k tzv. zoomování a je potřeba data v celém mapovém okně vykreslit znova.

Při pohybu mapového okna se mění pouze rozsah. Často se stává, že uživatel pohne s oknem jen o kousek, kdy značná část okna je pořád v rozsahu již stažených dat. V tomto případě je zde prostor pro optimalizaci, kdy by nebylo potřeba získávat z WMS serveru všechna data v rozsahu mapového okna, ale pouze ta data, která dosud nebyla stažena.

3 WMS modul pro SAGA GIS

Koncept modulu v SAGA GIS je odlišný od GRASS. Moduly v SAGA GIS mohou i za svého běhu komunikovat s uživatelem a reagovat na jeho podněty. Tyto vlastnosti lze velmi dobře využít při implementaci WMS.

3.1 Analýza původního stavu

Do aktuální verze (2.0.8) SAGA GIS byla nově začleněna experimentální knihovna *Garden - Web Service Data Access*. Tato knihovna obsahuje modul *Import a Map via Web Map Service (WMS)* umožňující získat data z WMS serveru. Modul funguje tak, že po spuštění vytvoří ze zadané URL adresy dotaz GetCapabilities a na základě odpovědi vytvoří dialog, ve kterém si uživatel může vybrat parametry pro získání rastru z WMS serveru.

Toto řešení WMS modulu je uživatelsky velmi přívětivé, jelikož stačí pouze znát URL adresu WMS serveru a ostatní kroky vedoucí k získání dat z WMS serveru jsou již v režii modulu.

Jelikož se jedná o experimentální modul v první verzi, má určité nedostatky. Mezi největší patří:

- Nezobrazí uživateli všechny možnosti, které server nabízí, protože špatně zpracovává Capabilities soubor. Z tohoto pohledu mezi největší chyby patří:
  - Modul z Capabilities souboru načte pouze ty vrstvy, které jsou přímými potomky kořenové vrstvy. Ostatní vrstvy jsou ignorovány.
  - Nelze určit pořadí vrstev, v jakém budou zobrazeny na výsledné mapě.
  - Neumožňuje výběr stylu vrstvy .
  - Nebere v úvahu dědičnost ve stromu vrstev.
- Nepodporuje dlaždicování.

3.2 Stanovení cílů implementace

Základní koncept modulu bude stejný jako u modulu experimentálního. To znamená, že modul nejprve získá Capabilites soubor z WMS serveru, vygeneruje pro něj

nabídku a na základě zvolených hodnot pomocí dotazu GetMap stáhne požadovaný mapový rastr a naimportuje jej do programu.

Při implementaci by měly být naplněny tyto cíle:

- Modul bude schopen správně zpracovat Capabilities soubor ve standardu 1.1.1 a 1.3.0. Jelikož ne všechny WMS servery zcela naplňují WMS standardy, bude napsán tak, aby nelpěl na absolutní implementaci WMS standardu serverem.
- Nabídka parametrů umožní uživateli využití všech možností WMS dotazu GetMap.
- Modul bude schopen transformovat rastr do uživatelem požadováného zobrazení. V SAGA GIS neexistuje ekvivalent lokace jako v GRASS. S vrstvami lze pracovat v libovolném zobrazení. Existuje i modul, který umí transformovat rastr do jiné projekce. Přesto tato funkcionalita může uživateli velmi ulehčit práci, protože se často stává, že WMS server neposkytuje data v uživatelem požadované projekci.

  Také modul umožní zadat hodnotu parametru `BBOX` v souřadnicích požadované projekce a za běhu modulu tyto souřadnice transformovat do projekce WMS dotazu.
- Umožní implementovat další rozšíření WMS standardu.

3.3 Volba způsobu implementace

Modul byl implementován v jazyce C++, stejně jako ostatní moduly v SAGA GIS. Byla snaha využít objektových rysů tohoto jazyka, které mohou ulehčit budoucí rozšiřování modulu o podporu dalších nástaveb WMS a celkově pomoci vytvořit čitelnější kód.

Důležitou volbou, před vlastní implementací modulu, bylo rozhodnutí, zda spouštět další SAGA moduly, které by například transformovaly rastr do požadované projekce nebo transformovaly souřadnice.

Nakonec bylo rozhodnuto tyto operace provést před importem do SAGA GIS pomocí knihovny GDAL (reprojekce) a knihovny PROJ4 (transformace souřadnic),

jelikož spouštění ostatních modulů z modulu není v SAGA GIS tak jednoduché jako v GRASS, kde je možné modul zavolat prostřednictvím jediné funkce.

SAGA moduly jsou kompilovány do dynamických knihoven, které jsou po stratu programu načteny. Problémem je nutnost uvedení cesty v souborovém systému, kde se nachází knihovna zahrnující spouštěný modul. Knihovny jsou instalovány do určeného adresáře, avšak nelze vyloučit, že uživatel může tyto knihovny přesunout do jiných adresářů a v tomto případě by nebylo možné modul spustit.

Knihovna PROJ4 [19] je open source projekt, který dovoluje transformovat souřadnice do různých projekcí, které jsou součástí této knihovny. Také umožňuje definovat vlastní projekce. Tuto knihovnu používá např. GRASS, SAGA GIS nebo GDAL.

3.4 Moduly v SAGA GIS

Každý modul v SAGA GIS je potomkem třídy `CSG_Modul`. Vstupní argumenty modulu, které se zadávají před spuštěním, jsou uvedeny v konstruktoru třídy modulu.

Argumenty jsou prvky instance třídy `CSG_Parameters Parameters`. Metoda `OnExecute` je další metoda, kterou musí každý modul obsahovat. Tato metoda je volána při spuštění modulu.

V SAGA GIS také existují interaktivní moduly, které jsou schopny reagovat např. na kliky v mapovém okně nebo na stisky kláves, jejichž struktura se liší.

3.5 Implementace modulu

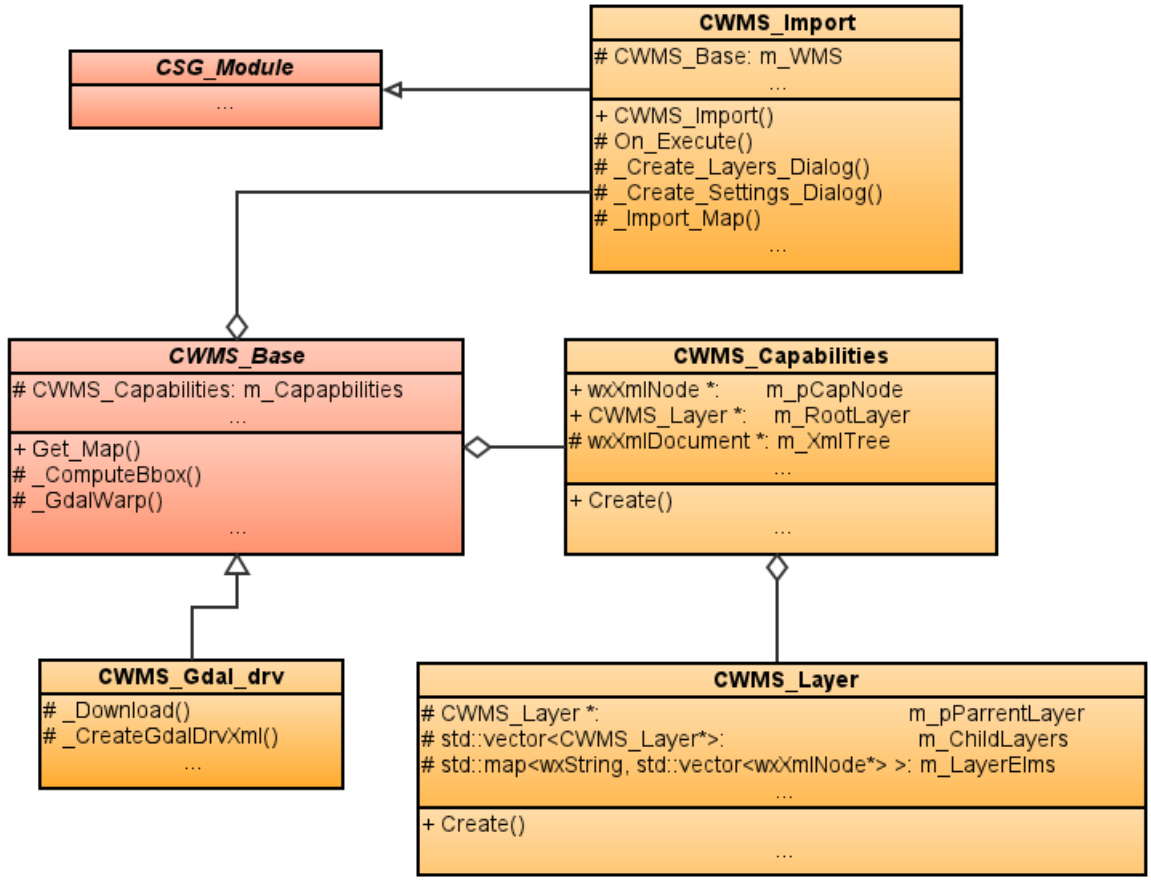
Modul je implementován jako přímý potomek třídy CSG_Module. Tato třída se nazývá `CWMS_Import` a má na starosti všechny funkce, které jsou přímo spojeny s programem SAGA GIS. To znamená, že tato třída využívá částí SAGA API. Ostatní třídy modulu jsou na SAGA API nezávislé.

`CWMS_Import` vytváří dialogy na základě Capabilities souboru, který je stažen a načten skrz členy třídy `CWMS_Base` a importuje rastr do programu. K těmto činnostem modul používá instance tříd definovaných v SAGA API.

[19] `http://trac.osgeo.org/proj/`

Jádrem modulu je metoda `OnExecute`, ze které jsou volány všechny ostatní prvky kódu modulu.

Struktura modulu je popsána v následujícím UML diagramu zahrnujícím nejdůležitější třídy a jejich významné metody a členy:



3.5.1 Třída `CWMS_Base`

Tato třída nejprve pomocí instance `CWMS_Capabilities` získá Capabilities data z WMS serveru. Na základě těchto dat instance `CWMS_Import` vygeneruje dvě nabídky, ze kterých uživatel vybere parametry dotazu GetMap.

Potom je spuštěna metoda `GetMap`, která stáhne a transformuje rastr do požadované projekce. Tato metoda vrátí cestu k souboru, ve kterém je uložena již výsledná rastrová mapa určená k načtení do programu. Struktura a funkce této metody jsou velmi

podobné WMS modulu pro GRASS. Největším rozdílem je to, že nenačítá výslednou mapu do programu, protože k tomu je potřeba využít neveřejných členů třídy `CSG_Module`, které nejsou z této třídy dostupné. Toto by šlo obejít deklarováním `CWMS_Base` přítelem `CWMS_Import`, což je potomek `CSG_Module`. Tato deklarace však porušuje zapouzdřenost, jeden ze základních principů objektově orientovaného programování. Proto je metoda `_Import_Map` součástí třídy `CWMS_Import`, z níž může přistupovat k těmto členům.

Metoda `_Import_Map` importující rastr do programu je jedinou částí modulu, která byla převzata z původního modulu a nebyla zásadně upravena.

Metoda `Get_Map` volá tři neveřejné metody v tomto pořadí:

- `_ComputeBbox` - Transformuje souřadnice obdélníku do projekce WMS dotazu. Transformace je provedena pomocí knihovny PROJ4, která je standardně přítomna v každé SAGA instalaci. Z transformovaných bodů jsou vybrány extrémní souřadnice, které pak definují parametr `BBOX` WNS dotazu.
- `_Download` - Tato metoda je virtuální a je pouze v této třídě deklarovaná. Návratovou hodnotou metody je cesta k souboru rastru, ve kterém jsou již spojeny dlaždice z jednotlivých WMS dotazů.
- `_GdalWarp` - Tato metoda pomocí knihovny GDAL transformuje stažený rastr do výsledné projekce. Jedná se o modifikovaný kód z `http://www.gdal.org/warptut.html`.

3.5.2 Třída `CWMS_Gdal_drv`

Členění této třídy a funkce metod jsou v zásadě stejné jako u třídy `WMSGDALDrv` v GRASS modulu. Metoda `_CreateGdalDrvXml` vytvoří XML soubor s parametry pro GDAL WMS ovladač, který metoda `_Download` použije pro stažení rastru z WMS serveru.

3.5.3 Třída `CWMS_Capabilities`

Tato třída pomocí metody `Create`, jejíž vstupním argumentem je URL adresa WMS serveru, vytvoří WMS dotaz GetCapabilities a následně načte do své struktury XML

soubor s odpovědí.

K načtení XML souboru jsou použity třídy knihovny WxWidgets [20], která je součástí programu SAGA. Pomocí třídy `wxXmlDocument` této knihovny je načten XML soubor a vytvořena datová struktura reprezentující strom elementů tohoto souboru. Ekvivalentem XML elementu v této struktuře je třída `wxXmlNode`, která obsahuje odkaz na rodiče a na své přímé potomky.

Pokud by neexistovala dědičnost ve stromu elementů `<Layer>`, bylo by možné snadno zjistit všechny informace o vrstvě z jejího `wxXmlNode` a stačilo by pouze uchovat ve třídě `CWMS_Capabilities` odkaz na kořenový element XML souboru. Protože však tato dědičnost existuje, bylo potřeba nějakým způsobem ke každé vrstvě seskupit všechny její elementy, včetně zděděných.

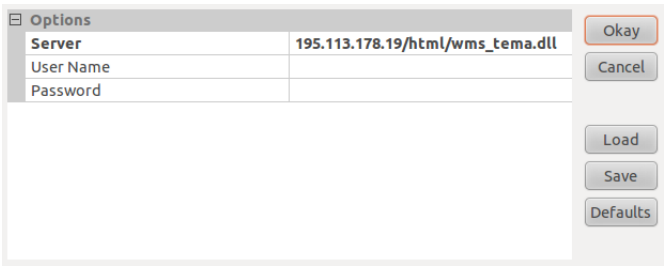
Proto byla vytvořena třída `CWMS_Layer`, která obsahuje ukazatel na rodičovskou vrstvu a na její přímé potomky ve stromu. Důležitým členem třídy je kontejner standardní knihovny C++ map s názvem `m_LayerElms`, který obsahuje položky mající jako klíč název elementu, který je přímým potomkem nebo elementem děděným elementem `<Layer>` a hodnotou tohoto klíče je vektor ukazatelů na objekty `wxXmlNode` reprezentující elementy s tímto názvem. Jelikož se jedná o ukazatele na objekty, nejsou ukládány duplicitní informace, ale všechny vrstvy, které dědí stejný element odkazují na tentýž objekt. Například pro elementy `<Style>` je v tomto kontejneru vytvořena nová položka, jejímž klíčem je text `Style` s vektorem, který odkazuje na všechny elementy `<Style>` prostřednictvím objektu `wXmlNode`.

Toto uspořádání je vytvořeno rekurzivně, kdy je strom elementů `<Layer>` procházen od kořene k jeho listům. Po vytvoření instance `CWMS_Layer` je zavolána její metoda `Create`, která z kontejneru `m_LayerElms` rodičovského objektu třídy `CWMS_Layer`, který je již díky rekurzi vytvořen, vybere ty objekty, které dědí a naplní vlastní kontejner `m_LayerElms`. Aby kontejner obsahoval všechny přímé potomky elementu `<Layer>`, jsou potom přidány elementy, které se nedědí.

[20] `http://www.wxwidgets.org/`

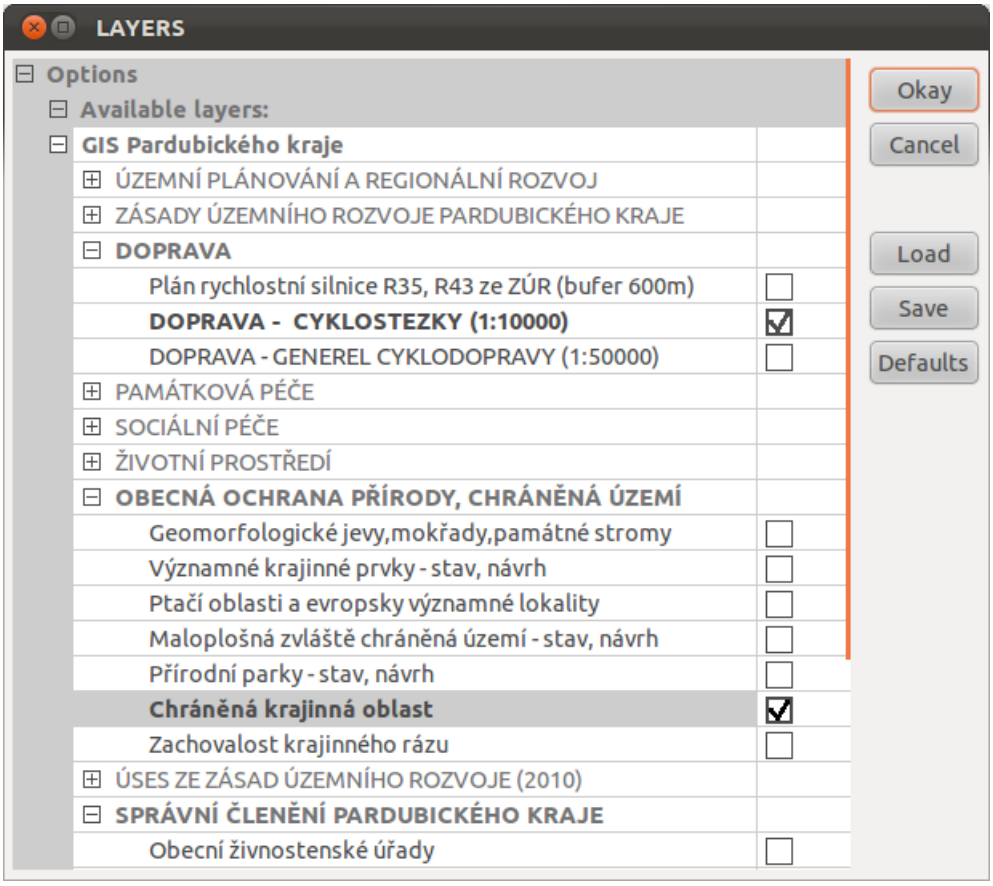
3.6 Ukázka práce modulu

Pokud je modul načten do SAGA GIS, je dostupný v knihovně modulů *Web Service Data Access* pod názvem *WMS*. Po otevření modulu se objeví toto dialogové okno:



V tomto případě byla zadána URL adresa WMS serveru Pardubického kraje. Pokud server vyžaduje uživatelské jméno a heslo, lze tyto parametry zadat do polí `User Name` resp. `Password`.

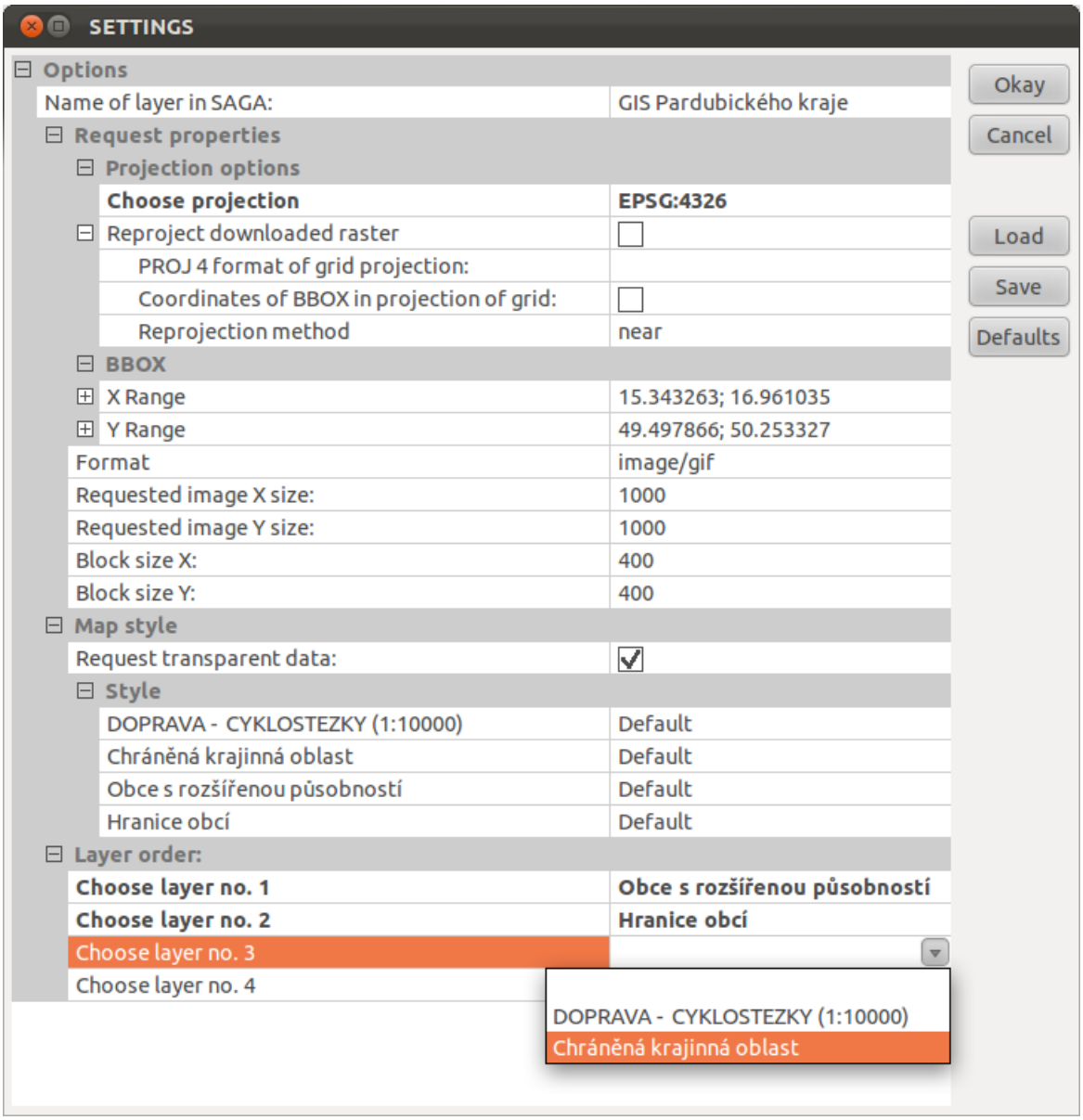
Jako další dialog se zobrazí strom vrstev, které server poskytuje:



Je možné vybrat pouze ty vrstvy, které mají definován uveden element `<Title>`. Jak bylo již výše zmíněno, vrstvy které tento element postrádají nelze použít pro

dotaz typu GetMap.

Posledním dialogem generovaným modulem je nastavení parametrů WMS dotazu. Pro tento příklad, kdy byly vybrány vrstvy `Obce s rozšířenou působností`, `Hranice obcí`, `DOPRAVA - CYKLOSTEZKY (1:0000)` a `Chráněná krajinná oblast` vypadá takto:

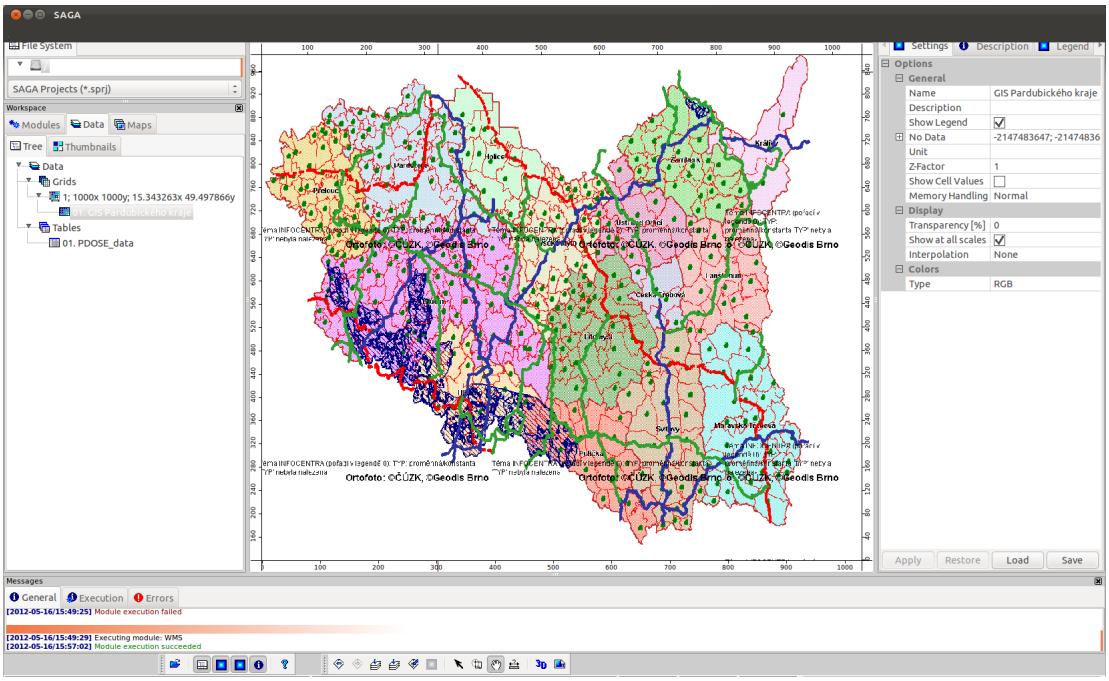


Dialog obsahuje tyto položky:

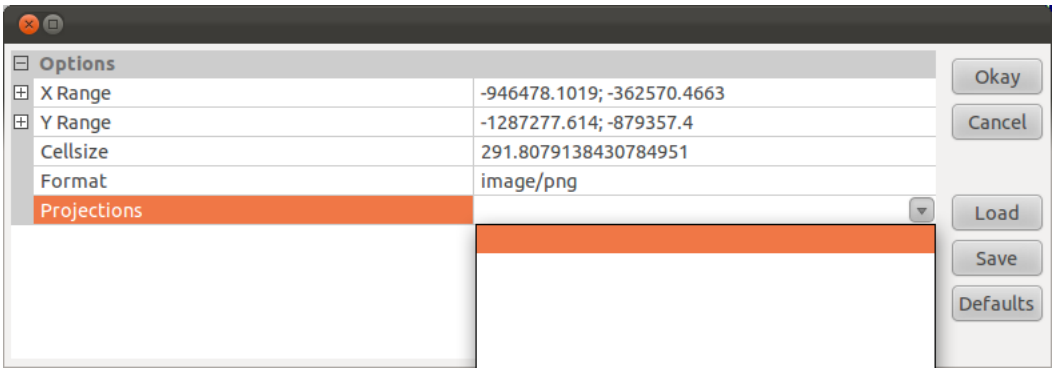
- `Name of layer in SAGA` - Představuje název vrstvy se staženými rastrem v SAGA.
- `Choose projection` - Tato nabídka obsahuje seznam všech společných projekcí, ve kterých jsou dostupné vybrané WMS vrstvy. Pokud uživatel změní vybranou projekci, jsou automaticky změněny hodnoty v oddíle `BBOX`, na hodnoty pro tuto projekci z elementu `<BoundingBox>`. Toto slouží k usnadnění orientace uživatele, v jakém rozsahu jsou data poskytována.
- Pokud uživatel chce rastr stažený z WMS serveru transformovat do jiné projekce, definuje tuto projekci v parametru `PROJ4 format of grid projection`. Pokud má být rastr transformován, musí také být zatržena volba `Reproject downloaded raster`. Jestliže jsou zadány souřadnice `BBOX` v projekci, do které rastr bude transformován, je potřeba zatrhnout volbu `Coordinates of BBOX in projection of grid`.
- Po oddíle `BBOX` jsou definovány rozměry staženého rastru v pixelech (`Requested image X/Y size`)a počet pixelů v řádcích a sloupcích jedné dlaždice (`Block size X/Y`).
- V oddíle `Map Style` volba `Request transparent data` udává, zda mají být požadovány data s průhlednou vrstvou. V pododdíle `Style` jsou zobrazeny nabídky se všemi dostupnými styly jednotlivých vrstev.
- V posledním oddíle je možné určit pořadí vrstev, v jakém budou budou vykresleny ve výsledné mapě. Aby si uživatel nemusel pamatovat, které vrstvy již vybral ve výše uvedených nabídkách, jsou již vybrané vrstvy z těchto nabídek odstraněny. Výběr pořadí vrstev by bylo lepší implementovat jako seznam položek, které by šlo přesouvat. Avšak SAGA API neumožňuje vytvořit takovýto prvek v dialogu modulu.

WMS server v tomto příkladě nepodporuje standard zcela správně. Problémem je, že vrstvy řadí v opačném pořadí, takže ta vrstva, která je vybraná jako první (Obce s rozšířenou působností), bude překryta ostatními vrstvami. Pokud by se WMS server choval správně, měla by být zobrazena nad všemi vrstvami.

Následně na základě zvolených parametrů modul stáhne data z WMS serveru a importuje je do SAGA GIS. V tomto případě stažená data vypadají takto:



Původní modul na tomto příkladě selže, protože nenabídne uživateli žádné vrstvy, jelikož v úvahu bere pouze vrstvy, které jsou přímým potomkem kořenové vrstvy (např. `DOPRAVA`) a v tomto příkladě jsou všechny tyto vrstvy bez elementu `<Title>`, čili nepoužitelné pro dotaz GetMap. Dialog, který modul vygeneruje v tomto příkladě vypadá takto:



Protože modul generuje pouze jeden dialog, měl by obsahovat i nabídku vrstev, které v tomto případě nenačte. Jak je vidět, dalším problémem je špatné načtení informací o projekci, kdy položky výběru jsou prázdné. Toto není jediný problém spojený s projekcemi. Protože modul načítá projekce rovněž z kořenového elementu, není schopen nabídnout, ty projekce které jsou uvedeny u vrstev níže ve stromu.

3.7 Další vývoj

Nejdůležitějším krokem, který by se měl udát v blízké době, bude zveřejnění modulu komunitě uživatelů SAGA, aby bylo možné modul řádně otestovat a upravit jej podle připomínek uživatelů. Způsob zveřejnění bude muset být ještě dohodnut s vývojáři SAGA.

Do modulu by měla být přidána nová třída `CWMS_Drv`, která bude přímo implementovat komunikaci s WMS serverem. Princip této implementace bude velmi podobný třídě `WMSDrv` v modulu GRASS. Neměl by být problém začlenit tuto třídu do stávajícího kódu, protože stávající struktura modulu je již k tomu přizpůsobena.

Modul by měl běžet ve vlastním vlákně, protože pří stahování dat se program SAGA odmlčí. Tento stav je velmi nepříjemný pokud stažení dat trvá delší dobu.

Dalším krokem by bylo provázání modulu s mapovým oknem, kdy by docházelo k dynamickému zobrazování WMS dat v mapovém okně na základě jeho aktuálního rozsahu.

K tomuto nelze využít stávajících možností interaktivních modulů, které mohou reagovat jen na akce vyvolané myší nebo klávesnicí.

4 Závěr

V rámci této bakalářské práce byla vylepšena podpora WMS v programech GRASS a SAGA GIS.

V programu GRASS byl vytvořen WMS modul, který odstraňuje problémy původního WMS modulu a byl navržen tak, aby byl lehce rozšiřitelný o podporu dalších nadstaveb WMS standardu, čímž by uživatelům GRASS umožnil přístup k zdrojům dat, které jim nyní zůstávají do velké míry zapovězeny. Při implementaci byly naplněny všechny stanovené cíle.

Také byly učiněny některé kroky v rámci integrace podpory WMS do GRASS GUI, které by měly v budoucnu vést ke srovnatelné práci s WMS daty v GRASS jako v programech ArcGIS nebo QGIS, které jsou v tomto špičkou mezi GIS aplikacemi.

V SAGA GIS byl experimentální WMS modul, který má velké nedostatky, přepracován a jeho funkcionalita rozšířena do takového stavu, že nyní je možné jej považovat za plnohodnotný WMS modul. Rovněž u tohoto modulu byly při implementaci naplněny všechny stanovené cíle.

Velmi zajímavou zkušeností byla práce na dvou různých open source projektech. Co se týče projektu SAGA GIS, je zde velmi znát, že jeho komunita je mnohem menší než komunita kolem GRASS. To se zejména projevuje na téměř neexistující dokumentaci pro vývojáře, a proto je potřeba většinu informací vypátrat přímo ve zdrojovém kódu.

Další velkou zkušeností byl vývoj v podstatě velmi podobných funkcionalit řešící stejný problém ve dvou odlišných programovacích jazycích Python a C++. Díky tomu jsem měl možnost si prakticky potvrdit známý fakt, že C++ je sice velmi silný nástroj a efektivní programovací jazyk, jehož odvrácenou stranou ve srovnání s jazykem Python je mnohem delší a pracnější vývoj.

Seznam použitých zkratek

API	Application Programming Interface
CGM	Computer Graphics Metafile
EPSG	European Petroleum Survey Group
GDAL	Geospatial Data Abstraction Library
GIS	Geographic Information System (Geografický informační systém)
GPL	General Public License
GRASS	Geographical Resources Analysis Support System
GUI	Graphical User Interface (Grafické uživatelské rozhraní)
HTTP	Hypertext Transfer Protocol
JPEG	Joint Photographic Experts Group
MIME	Multipurpose Internet Mail Extensions
PNG	Portable Network Graphics
QGIS	Quantum GIS
OGC	Open Geospatial Consortium
SAGA	System for Automated Geoscientific Analyses
SVG	Scalable Vector Graphics
TIFF	Tag Image File Format
UML	Unified Modeling Language
URL	Uniform Resource Locator
WMS	Web Map Service
WMTS	Web Map Tile Service
XML	Extensible Markup Language

Použité zdroje

[1] Open Geospatial Consortium *Web Map Service Implementation Specification - Version 1.3.0"* [online]. 2004 [cit. 2012-02-22]. URL:<http://portal.opengeospatial.org/files/?artifact_id=14416>

[2] Open Geospatial Consortium *Web Map Service Implementation Specification - Version 1.1.0"* [online]. 2002 [cit. 2012-02-22]. URL:<http://portal.opengeospatial.org/files/?artifact_id=1081&version=1&format=pdf>

[3] NETELER, Markus; MITASOVA, Helena. *Open Source GIS: A GRASS GIS Approach*. 3rd Ed. New York: Springer, 2008. 406 s. URL: <http://www.grassbook.org>. ISBN 978-0-387-35767-6.

[4] RAPPIN, Noel; DUNN, Robin. *WxPython in Action*. Greenwich, USA: Manning Publications Co., 2006. 552 s. ISBN 1-932394-62-1.

[5] SAGA User Group Association. *SAGA API* [online]. 2011 [cit. 2012-03-22]. URL: <http://www.saga-gis.org/saga_api_doc/html/index.html>

[6] GRASS Development Team. *GRASS 7 Programmer's Manual* [online]. 2012 [cit. 2012-03-02]. URL: <http://www.saga-gis.org/saga_api_doc/html/index.html>

[7] *GRASS GIS Tracker and Wiki* [online]. 2012 [cit. 2012-03-01]. URL: <http://trac.osgeo.org/grass>

[8] GEPRO spol s.r.o. *WMS služby v ČR* [online]. 2011 [cit. 2012-02-19]. URL: <http://www.gepro.cz/geodezie-a-projektovani/tipy-a-triky/wms/wms-sluzby-v-cr/>

[9] SKYLAB Mobilesystems Ltd. *OGC WMS Server List* [online]. 2009 [cit. 2012-02-19]. URL: <http://www.skylab-mobilesystems.com/en/wms_serverlist.html>