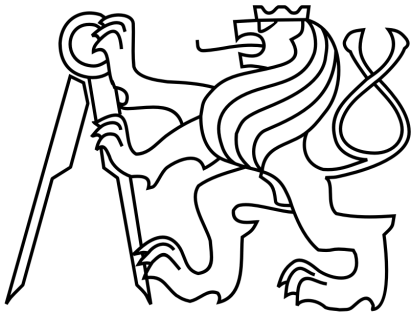
České vysoké učení technické v Praze

fakulta elektrotechnická

Bakalářská práce

Vzdálené generování Ganttových diagramů



Vypracoval : Radovan Černý

Vedoucí : Ing. Michal Kutil

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídící techniky Školní rok 2005/2006

Zadání bakalářské práce

Student Radovan Černý

Obor: Kybernetika a měření

Název tématu: Vzdálené generování Ganttových diagramů

Zásady pro vypracování:

1) Prostudujte technologii distribuovaných systémů Web Services

2) Naimplementujte serverové rozhraní poskytující přístup k nástroji generujícímu Ganttovy diagramy

3) Vytvořte klientskou aplikací využívající uvedené serverové rozhraní

Vedoucí bakalářské práce: Ing. Michal Kutil

Datum zadání bakalářské práce: Únor 2006

Termín odevzdání bakalářské práce: 30.6.2006

!!! Zde bude originál !!!

Prohlášení o autentičnosti

Já, Radovan Černý, prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady v práci uvedené.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V praze dne: 20.7.2006 Podpis: ……

Poděkování

Poděkování na tomto místě patří hlavně panu Kutilovi, který nad mou prací dozíral a trpělivě směroval mou činnost zkušenými radami.

Abstrakt

Cílem mé práce bylo naimplementovat Gantt generátor, který by využíval skript Gantt a umožnil uživateli vzdáleně generovat ganttovy diagramy. Tyto mělo být možné generovat z webového rozhraní a z Matlabu. Vzhledem k požadované architektuře klient-server jsme pro komunikaci zvolili WebServices.

Výsledkem mé práce tak jsou skripty, které mohou rozšířit TORSCHE Scheduling Toolbox vyvíjený na katedře řídící techniky.

Abstract

The aim of my work was to implement Gatnn chart generator that would use skript Gantt and let user remotely generate Gantt charts. These charts should be possible to be generated from web-interface and from Matlab. Taking into consideration the necessity of using client-server architecture we desided to use WebServices for communication through the internet.

The result of my work are scripts that can be used to expand TORSCHE Scheduling Toolbox that is developed on the Department of Control Engineering.

Obsah

Seznam obrázků a ukázek kódů

Seznam obrázků:

Seznam kódů:

1 Úvod

1.1 Rozbor zadání

1.1.1 Vzdálené generování:

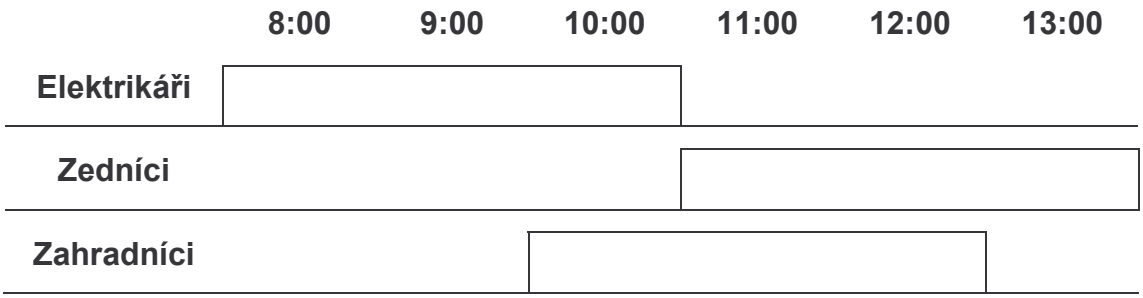
Vzdálené generování je proces, kdy přes internet spouštíme proceduru umístěnou na jiném počítači, předáme ji vstupní data a ona vygeneruje data výstupní. V našem případě jsou vstupem jeden až dva textové soubory s popisem požadovaného Ganttova diagramu. Výstupem je Ganttův diagram.

Pro realizaci toho postupu jsme zvolili architekturu Klient-Server ve spojení s technologií WebServices.

1.1.2 Ganttův diagram

Ganttův diagram je vlastně časovým rozvrhem činností, prací, úkonů, v němž je zachyceno několik paralelně činných jednotek, jejichž konání na sebe může navazovat a je tedy nutné dodržet pořadí prací.

Příkladem *Ganttova diagramu* může být situace na stavbě. Představme si, že máme skupinu elektrikářů, zedníků a zahradníků. Víme, že před vyrovnáním zdí je nutné mít elektriku hotovou. Naopak zahradníci nejsou na nikoho vázání (počítáme-li s tím, že elektrika je přes pozemek již zavedena). Potom by diagram prací mohl vypadat následovně:



Obrázek 1.1) Ganttův diagram

1.2 Účel mé aplikace, Scheduling Toolbox

Má aplikace bude použita pro balík nástrojů nesoucí název TORSCHE Scheduling Toolbox for Matlab [10], TORSCHE.

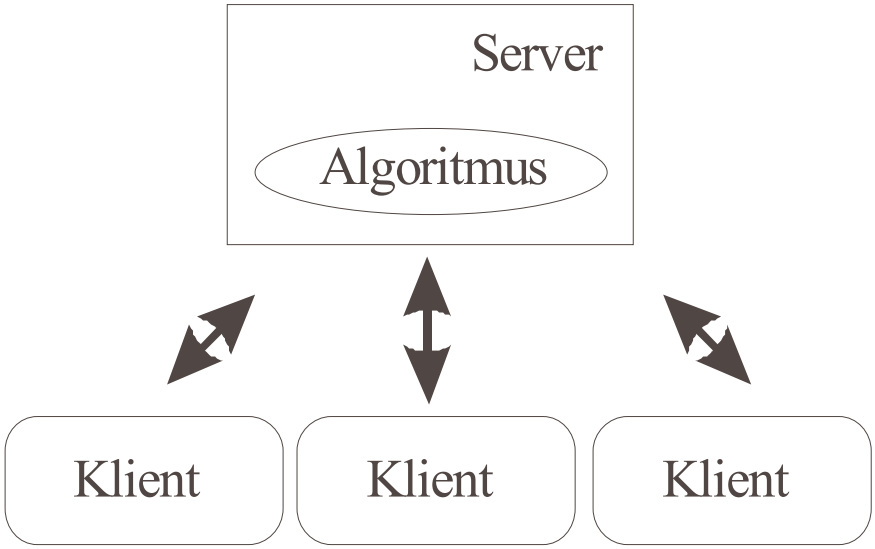
TORSCHE (**T**ime **O**ptimisation of **R**esources, **SCHE**duling) je toolbox poskytující různé plánovací algoritmy, které se dají použít například pro plánování paralelních procesů, pro optimalizaci produkce na výrobních linkách atp. Použitím tohoto toolboxu lze jednoduše získat optimální kód pro intenzivní výpočty běžící na specifickém hardwaru. Největší přínos tohoto balíčku, postaveného na teorii grafů a operačním výzkumu, je v tom, že umožní použití svých funkcí pro široké spektrum problémů.

TORSCHE nabízí kolekci Matlabovských metod, které umožní uživateli zformovat plánovací problém během rozvažování odpovídající konfigurace zdrojů, parametrů úloh a optimalizačních kritérií. Řešení problémů probíhá buď jejich přeformulováním, nebo přímým řešením během vybírání odpovídajícího plánovacího algoritmu. Vstupní data jsou většinou ve formě orientovaného grafu a výstupem je tak rozvrh, který může být zobrazen pomocí ganttova diagramu.

2 Použité technologie

2.1 Klient-Server

Tato architektura dělí své účastníky na klienty a server, kde server je centrální jednotka, která dává k dispozici metody a služby. Ty potom volá klient. Server také může data jen distribuovat, podobně jako router. Oba mohou, ale nemusí, běžet na stejném hardwaru. Komunikace mezi nimi probíhá například pomocí WebServices.



Obrázek 2.1) Architektura Klient-Server

2.2 WebServices

2.2.1 Základní části Web Services

WebServices - webové služby - mají za cíl zpřístupnění zdrojů snadno zpracovatelných strojově, tj. umožnit snadnou spolupráci programů běžících na libovolných platformách (programovací jazyk, CPU, OS, mobilní telefon). Jsou založené na standardech *World Wide Web Consorcia (W3C)*.

WebServices je technologie pro vzdálené volání procedur *(RPC - Remote Procedure Call)* pomocí přenosu zpráv zapsaných v jazyku XML, které se posílají pomocí protokolu HTTP.

Pro definici typů dat využívá standard XML Schéma (struktura a typ předávaných dat). Pro identifikaci objektů (dat, jmen operací, atd.) využívá standard XML Namespaces (lze zamezit kolizím stejných jmen pro různé věci). O standardu XML schématu a XML namespaces se zmíním v další kapitole.

Technologii webových služeb tvoří tři části:

- **SOAP**: protokol pro vzdálené volání procedur
- **WSDL**: jazyk pro popis poskytovaných služeb
- **WSIL, UDDI**: mechanismy pro nalezení služeb

2.2.1.1 SOAP

Vývoj protokolu SOAP (Simple Object Access Protocol) započal předáváním parametrů metodou GET, dále pokračoval přes metodu POST, která umožnila k textovým parametrům přidat soubor, a která byla později využita pro zasílání dat ve formátu XML. Dále se připojily XML Schéma a XML Namespaces a doplnila se informace jakou funkci voláme. Výslednému formátu zprávy říkáme SOAP.

Implementace tohoto protokolu je vyvíjeny různými výrobci a pro různé platformy, ale vše je vzájemně kompatibilní. Pomocí tohoto protokolu si předávají data např. klient a server. Veškeré informace jsou posílány ve formátu XML a to ve zprávách přenášených pomocí HTTP protokolu.

Abychom mohli s využitím protokolu SOAP něco volat, musíme nejdříve vědět co a jak volat. K tomu slouží další část Web Services, a to jazyk WSDL.

2.2.1.2 WSDL

WSDL (Web Services Description Language) je jazyk sloužící pro popis rozhraní webových služeb pomocí XML formátu, který zahrnuje kompletní, strojově zpracovatelnou dokumentaci služby. Díky WSDL tak víme jména dostupných operací, typů jejich parametrů a návratových hodnot. Také se pomocí něj dozvíme kde a jak je služba dostupná a správnou syntax volání operací.

2.2.1.3 WSIL a UDDI

WSIL a UDDI mají stejný účel, kterým je nalézání veřejně dostupných webových služeb. Každý z těchto dvou mechanizmů však volí jinou metodu hledání.

UDDI (Universal Description, Discovery and Integration) nabízí veřejnou databázi, kam poskytovatelé zapisují informace o svých službách, uživatelé ji mohou prohledávat a nalezené funkce použít. Tato databáze však nezaručuje důvěryhodnost poskytovatelů ani funkčnost (platnost) záznamů. To proto, že jednou zapsaná služba nemusí být stále funkční, podporovaná a dostupná. Odkaz tak nemusí být vždy použitelný.

Tuto službu lze přirovnat k databázi odkazů na běžných vyhledávacích serverech (např. www.seznam.cz)

WSIL (Web Services Inspection Language) má jiný mechanismus. Nejprve si vybereme dobrého a důvěryhodného poskytovatele, který nám dá popis svých služeb a rozhraní. Tento popis je uveden na známém místě a každý si jej tak může prohlédnout. Výhodu oproti UDDI to má v tom, že každý poskytovatel zpracovává svou vlastní databázi dostupných služeb, takže je i jeho přáním, aby databáze odpovídala skutečnosti.

Ani jeden z uvedených mechanizmů, tedy WSIL ani UDDI, však není standardem W3C. Více o WebServices lze nalézt na [15].

2.2.2 Výhody a nevýhody WebServices

Výhody Web Services jsou dány tím, že se data přenášejí pomocí XML, tedy v textové podobě a jsou nezávislá na platformě. Díky kódování UTF-8/16 nejsou ani problémy s češtinou.

Dalším kladem je, že o sobě klient a server vzájemně nic nepředpokládají. Web Services jsou mimo jiné i jednoduché a otevřené. Naopak nevýhodou zatím je nepříliš propracovaná bezpečnost.

2.2.3 Bezpečnost a WebServices

WebServices bezpečnost a autentizaci primárně neošetřují, ale protože potřeba chránit data je reálná, vznikly dva přístupy k zabezpečení.

2.2.3.1 Transport level security

Bezpečnost na úrovni přenosu se zajišťuje použitím SSL pod HTTP, tj. použitím HTTPS protokolu pro přenos SOAP zpráv. Nevýhoda je, že bezpečnost je zajištěna pouze mezi komunikujícími konci, a nelze například zpětně dokázat, že druhý konec poslal určitou zprávu, nebo u vícevrstvých aplikací nelze zjistit že zpráva nebyla změněna mezilehlými vrstvami. Tento způsob se používá a není složitý.

2.2.3.2 Message level security

Existují pokusy využít standardy XML Encryption a XML Signature pro šifrování a podepisování SOAP zpráv, protože SOAP zprávy jsou pouhé XML dokumenty. IBM si vydala vlastní WS-Security specifikaci. To samé udělal i Microsoft a jeho specifikace vystupuje pod jménem WS-SecureConversation. Jsou však snahy vše sjednotit do GT3 Security, ale zatím jde spíše o experiment.

Výhody bezpečnosti na úrovni zpráv by byly právě možnost zpětného prokazování co kdo poslal, a vyloučení mezilehlých vrstev z bezpečnosti komunikace. Naopak nevýhodou je vysoká režie zpracování.

2.3 Gantt

Gantt [8] je perlovský skript, který doplňuje Scheduling Toolbox a je klíčový pro mou práci, která využívá jeho funkce. Vstupem tohoto programu je název XML souboru. XML dokument obsahuje předpis pro vytvoření ganttova diagramu, tedy jména úloh, jejich časovou náročnost, začátek úlohy atp. Na základě tohoto popisu je vygenerován diagram, jehož vzhled lze upravit přiložením konfiguračního CNF souboru, který nese informace o změně jmen úloh, dále umožňuje určit měřítko os, barvu jednotlivých úloh v diagramu atd. Podmínkou je, aby se XML a CNF soubor jmenovaly stejně a lišily se jen příponou.

Výstupem skriptu je 6 souborů různých typů nesoucích stejné jméno jako XML soubor. Mezi tyto soubory patří MetaPost a EPS obrázek diagramu, dále pomocný MetaPost soubor s příponou mpx, log soubor, pak jeden soubor bez přípony a nakonec vznikne i nový CNF soubor odpovídající podobě diagramu. Pokud jsme spolu s XML dokumentem skriptu gantt odeslali i CNF soubor, bude nově vzniklý konfigurační soubor totožný a uživatel tak nic nezaznamená. Pokud jsme CNF nepřiložili, vytvoří se nové CNF na základě XML.

Gantt se skládá z těchto souborů:
- gantt.mp
- gantt.pl
- Gantt.pm
- TaskBox.pm

Program je dílem studenta Antonína Karlolíka (karolik@seznam.cz) a vznikl v letech 2005 a 2006, přičemž jej autor stále zdokonaluje.

2.4 XML

2.4.1 Co to je XML

XML (eXtensible Markup Language - rozšiřitelný značkovací jazyk) je metajazyk určený především pro obsahový popis dokumentu (popisuje se jen obsah - "datový model", nikoliv, jak se má dokument zobrazovat). Značky si můžeme definovat svoje vlastní, jejich interpretaci (zobrazení na WWW) pak určují ostatní technologie jako CSS. Síla jazyka XML spočívá ve skutečnosti, že jednotlivé prvky dokumentu jsou jednoznačně identifikovatelné a lze se na ně přesně odkazovat.

2.4.2 Standard XML Schéma

XML [13] je vlastně jedním z několika jazyků sloužících pro popis struktury XML dokumentů, jimiž si informačními systémy vyměňují strukturovaná data. Všechna schémata podporují datové typy, namespaces a syntaxi založenou na XML. Standard XML schéma je však nejvýznamnější, pochází od W3C [12] a je podporován významnými firmami.

2.4.3 Standard XML Namespaces

XML Namespaces [14], neboli jmenný prostor, definuje logické prostory jmen v XML dokumentu. Logickému prostoru odpovídá jeden celosvětově jedinečný identifikátor URI. Vyloučí se tím tak možnost, že se bude více proměnných jmenovat stejně.

3 Implementace

Požadavkem je vytvořit aplikaci typu klient-server, která bude umožňovat uživateli tvorbu ganttových diagramů z webového a Matlabovského prostředí. Tyto budou generovány skriptem Gantt umístěném na serveru *rtime.felk.cvut.cz*, jemuž mají mé skripty zprostředkovat data v podobě vstupního XML a případně i CNF souboru a to pomocí technologie WebServices. Výstupem serveru má být ganttův diagram v podobě PDF dokumentu a PNG obrázku. Dále je požadováno umožnit volbu jen některých výstupních souborů tak, aby nebyl uživatel nucen vždy přijímat například dokument PDF. V případě PNG formátu obrázku je navíc požadováno mít možnost určit jeho velikost.

Pro umožnění volby požadovaných výstupních souborů serveru jsou uživateli k dispozici funkce, jejichž návratová hodnota je patrná z názvu.

- `xml2cnf` na základě pouhého XML dokumentu vygeneruje defaultní CNF dokument, jehož následnou úpravou může uživatel vzhled diagramu měnit.

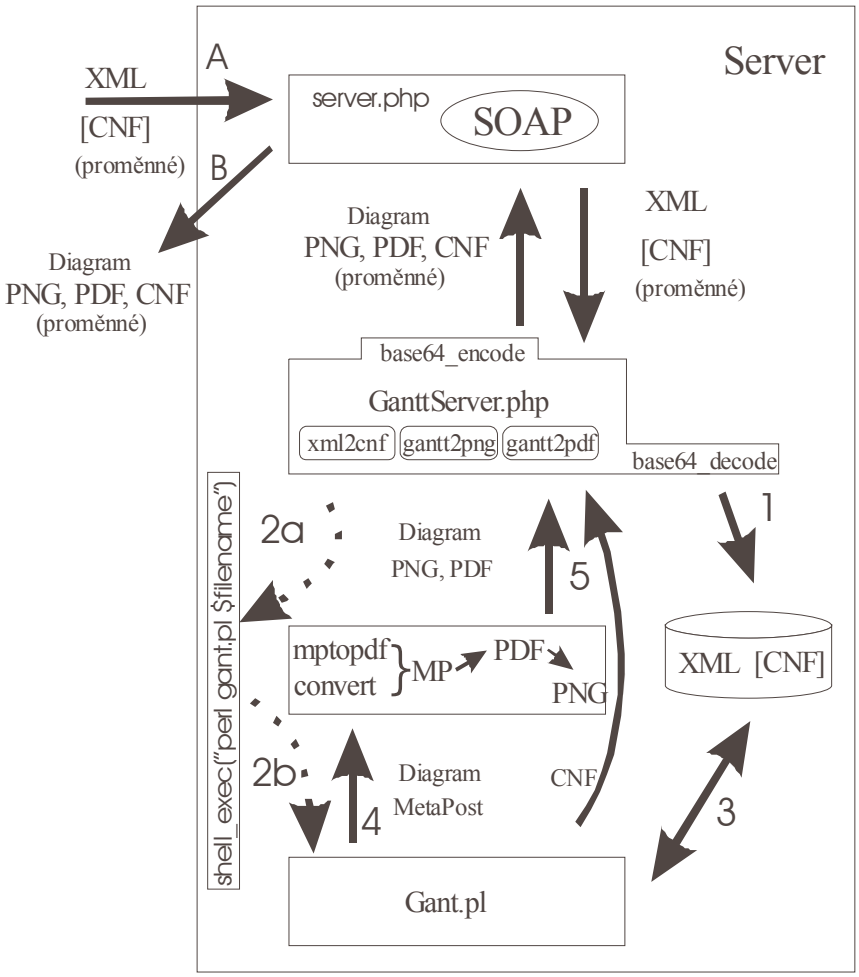
- `gantt2png` ze získaného XML, případně i CNF souboru, vygeneruje diagram požadovaného vzhledu a to ve formátu obrázku PNG.

- `gantt2pdf` má podobnou funkci jako `gantt2png` s tím rozdílem, že vrací PDF dokument.

3.1 Implementace serveru

3.1.1 Schéma funkce serverového skriptu

Server [2], jehož mnou navržené metody jsou v souboru GanttServer.php, má za úkol převzít od klienta vstupní data. Těmi jsou, mimo dodatečné parametry, hlavně CNF a XML soubory. Server soubory přijme a dá je k dispozici skriptu gantt.pl, který na jejich základě vygeneruje ganttův diagram a nový CNF soubor. Výstupem serveru pak bude diagram v podobě obrázku PNG a dokumentu PDF. Dále server vrátí nový CNF soubor. Pro přenos dat používám kódování base64 [6].



Obrázek 3.1) Schéma práce serveru

Protokol SOAP (viz Obrázek 3.1) slouží pro výběr a zavolání požadované funkce. Po obdržení dat zakódovaných v XML zprávě je nutné je rozkódovat a předat odpovídající funkci. Při vracení výstupních dat klientovi se opět vše zakóduje do zprávy ve formátu XML a odešle. Vše co je mimo oblast „server“ je ve formě XML zprávy. Označení šipek A, B je jen pro lepší orientaci ve vztahu schémat serveru a klienta (viz dále). Čísla u šipek naznačují pořadí vykonávání příslušných funkcí.

Server se skládá kromě souboru GanttServer.php též ze souborů server.php a functions.php, využívá knihovnu SOAP [7] a poskytuje tři základní funkce: xml2cnf, gantt2png, gantt2pdf. Server.php je základním souborem pro WebServices a pro generování WSDL dokumentu, který klientovi oznámí, jaké metody jsou dostupné, s jakými parametry se mají volat atp. WSDL dokument se získá zavoláním tohoto souboru s parametrem „wsdl“, tedy takto: server.php?wsdl.

Posledním souborem je soubor functions.php, který obsahuje pomocné funkce sloužící například pro ukládání souborů na disk, či pro zakódování dat zmíněným algoritmem base64.

3.1.2 Popis implementace serveru

Pro chod aplikace je nutné mít na serveru nainstalovanou knihovnu SOAP a zahrnout její soubory do php skriptu spolu se souborem functions.php. Soubory zahrneme příkazem require_once [5].

```
require_once 'SOAP/Value.php';
require_once 'SOAP/Fault.php';
require_once 'functions.php';
```

Kód 3.1) Require pro Server

Dále se musí každá funkce v tomto souboru, mimo klasické definování klíčovým slovek function, definovat pro potřeby WebServices i takto:

```
$this->__dispatch_map['xml2cnf'] =
array('in'=>array('XML'=>'int'),
            'out' => array('outputInt' => 'int'),
            );
```

Kód 3.2) Příklad deklarace funkce na serveru

Prvním z použitých algoritmů je uložení vstupních textových (XML, CNF) souborů, zakódovaných pomocí base64. Toto ukládání obsluhuje následující funkce:

```
function decodeString2File($string,$filename)
{
        $fp = fopen($filename,'w');
        fwrite ($fp,base64_decode($string));
        fclose(fp);
}
```

Kód 3.3) decodeString2File

Jména souborů, které na serveru ukládám musí být unikátní, aby nedocházelo ke kolizím. Takovéto jméno vygeneruji funkcí tempnam [5] a pro jasné rozlišení od jiných souborů na jeho začátek dávám jednotný řetězec, prefix „GanttServerTmp_“. Ten funkci zadám jako parametr. Dalším parametrem této funkce je cílová složka, ve které má mít soubor unikátní jméno.

```
$fileName = tempnam($folder, $prefix)
```

Kód 3.4) Tempnam

Oba vstupní soubory (XML,CNF) musí být uloženy do složky, ve které je skript pro tvorbu ganttových diagramů, gantt.pl. Po jejich uložení je již možné skript spustit a to pomocí funkce shell_exec [5]. Tato funkce umožní spustit shellovský příkaz.

```
shell_exec ("perl gantt.pl $fileName“)
```

Kód 3.5) Shell_exec (gantt)

Jediným parametrem této funkce je řetězec s daným příkazem. Tento řetězec se dále člení. První slovo, perl, je příkaz pro volání Perlu. Dále je uveden název spouštěného skriptu a nakonec proměnnou $fileName skriptu gantt.pl určíme název zdrojových souborů pro tvorbu diagramu.

Výsledkem činnosti skriptu gantt.pl bude vytvoření několika souborů různých typů: MetaPost (mp), konfigurační soubor (cnf), Encapsuled PostScript (eps) a dále soubory typu log a mpx.

Požadavkem však bylo, aby na výstupu měl uživatel k dispozici obrázek ve formátu PNG a PDF. K převodu na PDF poslouží funkce mptopdf, která převede MetaPost, v němž je vygenerovaný diagram, na PDF dokument. Tuto funkci je třeba opět spustit jako shellovský příkaz.

```
shell_exec('mptopdf MP 2>&1')
```

Kód 3.6) Shell_exec (mptopdf)

V předvedeném příkazu jen schématicky uvádím MP na místo názvu MP souboru. Ze vzniklého PDF již bude snadné vyrobit obrázek ve formátu PNG funkcí convert [4].

```
shell_exec("convert PDF PNG")
```

Kód 3.7) Shell_exec (convert)

Opět jsem pro zjednodušení uvedl jen schématické naznačení zadání jména PDF a PNG, tedy vstupního a výstupního, souboru.

Tento postup tvorby žádaných formátů diagramu jsem zvolil proto, že vytvořit PNG obrázek ze souboru EPS (který je jedním z výstupů skriptu gantt.pl a s nímž by měla funkce convert umět pracovat) se nedařilo kvůli fontům. Náš soubor EPS totiž neobsahuje přímo fonty, ale jen odkazy na ně. S nimi si funkce convert neporadila. Převod se však podařil z MetaPostu funkcí mptopdf. Její výstup, tedy

PDF dokument, v sobě fonty již obsahuje, proto se je povedlo promítnout i do PNG obrázku generovaného funkcí convert.

V této fázi tedy máme na základě vstupního XML, případně i CNF souboru vygenerovány výsledné PNG a PDF soubory spolu s novým CNF souborem, který vznikl zcela automaticky při běhu skriptu gantt.pl. Nyní je třeba vše zaslat zpět klientovi, opět pomocí kódování base64. To provede funkce encodeFile2String:

```
function encodeFile2String($filename)
{
	$fd = fopen ($filename, "r");

	$contents = fread ($fd, filesize($filename));
	fclose ($fd);

	$contents = base64_encode($contents); // $contens je string

	return $contents;
}
```

Kód 3.8) encodeFile2String

Server klientovi tedy nakonec vrací zakódovaný string. Nejobsáhlejší výstup mého serveru tedy je CNF soubor spolu s PNG obrázkem a PDF dokumentem. Pro každý z těchto 3 výstupů existuje jedna funkce příslušného jména (xml2cnf, gantt2png, gantt2pdf), která vrátí požadovaný soubor.

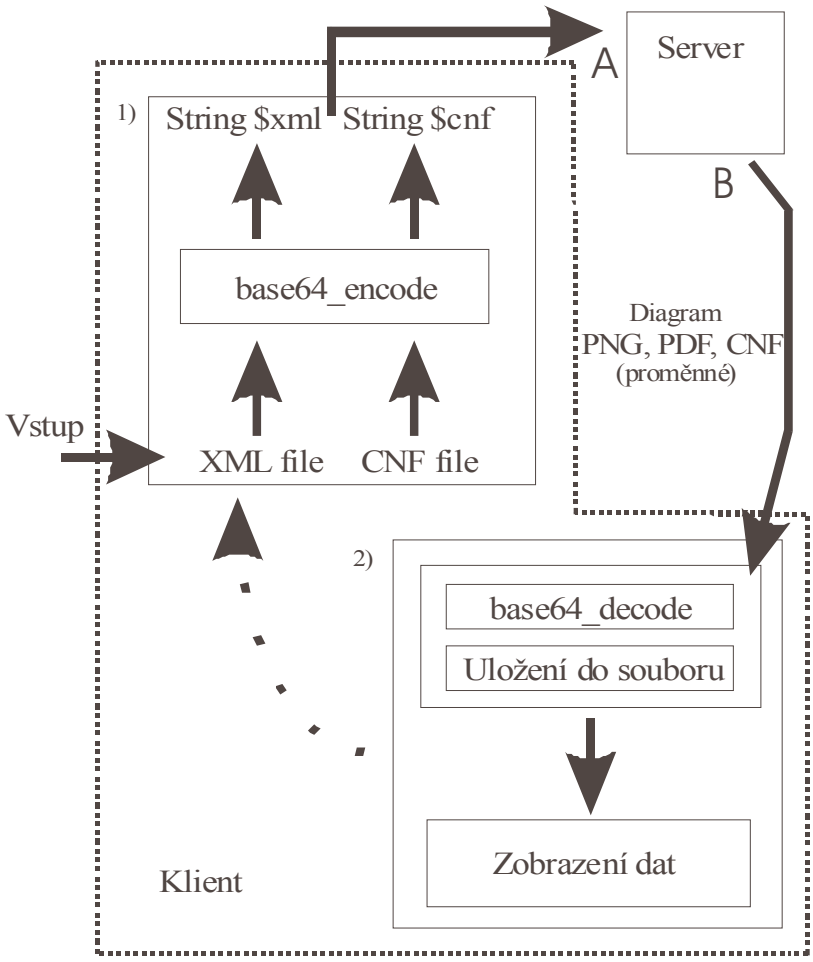
Pro tvorbu PNG obrázku je nutné funkci gantt2png zadat proměnnou pngdpi, která, zjednodušeně řečeno, určuje velikost vygenerovaného PNG obrázku. Jak jsem psal výše, PNG soubor se generuje z PDF dokumentu, který je vektorový. Pro generování PNG dokumentu je tedy nutné PDF dokument převést s určitým DPI, které určuje právě proměnná pngdpi. Čím vyšší DPI bude, tím větší bude PNG obrázek. Defaultně je pro funkci convert určeno 72 DPI, což bylo málo. Obrázek byl malý a při jakémkoli zvětšení nebyl hezký, proto pro náhled doporučuji 150 DPI.

3.2 Implementace klienta

3.2.1 Klient - PHP

3.2.1.1 Schéma funkce klientského skriptu

Mnou navržený klientský skript má za úkol si v prvním kroku vyžádat od uživatele vstupní data, ta zakódovat tak, aby je bylo možné odeslat serveru a spustit serverovou metodu s parametry v podobě zadaných XML a CNF souborů. Druhým krokem je přijmutí výsledných souborů od serveru, a to ve formátu PNG, PDF, či CNF a vše zobrazit uživateli tak, aby mohl diagram lehce upravit.



Obrázek 3.2) Schéma práce klienta

Položka „Vstup“ ve schématu klientského skriptu (Obrázek 3.2) představuje uživatele zadávajícího vstupní XML a CNF soubory pro tvorbu diagramu prostřednictvím webového browseru. Tečkovaná šipka pak značí možnost opětovné editace vygenerovaného diagramu a to úpravou souboru XML, nebo CNF. Klient se nachází na adrese *http://rtime.felk.cvut.cz/~cernyr1/gantt/index.php*

Klientská část aplikace se skládá z několika souborů. Nejdůležitějším je index.php, ve kterém je implementován celý klient. Dále je důležitý soubor _download.php ve složce GanttOutput_PhpClient, který slouží pro stahování výstupních PNG a PDF souborů. Další soubory obsahují například nápovědu (help.php), případně slouží pro tvorbu designu (kaskady.css, cnf_pic.png, xml_pic.png). Poslední dva soubory (demo.txt a demo.xml) jsou ukázkovými well-formed dokumenty, přičemž demo.txt je konfigurační soubor CNF, jehož příponu jsem změnil na „txt“ pro umožnění jeho prohlížení. Jsou dostupné přes nápovědu, či spuštěním demo projektu.

3.2.1.2 Popis implementace klienta

Protože využíváme WebServices založené na SOAP, musíme na začátku zahrnout jeho soubory do skriptu index.php, dále je nutné určit serverový skript a základní parametry, které jsou nutné pro jeho spuštění.

```
require 'SOAP/Client.php';
$soapclient = new
SOAP_Client('http://rtime.felk.cvut.cz/~cernyr1/gantt/server.php');
$options=array('namespace'=>'urn:Gantt_Server','trace'=>1,'timeout'=>1000);
```

Kód 3.9) SOAP na klientovi

Dále si pro uložení PDF a PNG souborů obdržených od serveru předem definujeme unikátní jména. A to funkcí tempnam [5], která pracuje stejně jako u serveru. Pro ukládání zmíněných souborů používám složku s názvem

GanttOutput_Php, proto její název této funkci předám parametrem spolu s prefixem „PhpClientTmp_“, který bude společný všem vygenerovaným názvům.

```
$tmpfname = tempnam('GanttOutput_Php/', 'PhpClientTmp_')
```

Kód 3.10) Tempnam

Při prvním načtení klientské stránky [1] se uživateli zobrazí prázdný formulář pro vkládání XML a CNF souborů, ve kterém má možnost zvolit si požadované výstupní soubory a velikost PNG obrázku. Pokud je zadán alespoň XML dokument, pokračuje skript po odeslání dvěmi možnými cestami. Byl-li soubor zadán do textové oblasti, tedy ručně vepsán, bude tento text po odeslání jednoduše dostupný v globálním poli $_POST a to rovnou v jediné proměnné. Pokud však byl soubor vložen prostřednictvím prvku „file“, tedy vybrán z disku, bude se soubor po odeslání nacházet v defaultním tempu a bude dostupný polem $_FILES. Zpracování obou variant proběhne následovně:

```
$newXMLText = stripslashes($_POST['XMLtext'])
```

```
$newXMLText = implode("",file($_FILES['MyXMLName']['tmp_name']));
```

Kód 3.11) XML to String

První ukázka kódu zobrazuje získání XML dokumentu vepsaného do textové oblasti z globálního pole $_POST. Funkce stripslashes [5] odstraní nežádoucí znaky, které byly automaticky přidány při odesílání textu.

Druhý kód ukazuje postup pro vložení celého odeslaného souboru prvkem „file“ do jediné proměnné. Samotný uploadovaný soubor je dostupný již zmíněným polem $FILES[$NazevPrvkuFileVeFormulari][´tmp_name´] (tmp_name je klíčové slovo pro vyvolání dočasného souboru, který je obrazem souboru odeslaného). Takto získaný soubor můžeme načíst řádek po řádku do pole funkcí file [5] a po té buňky

sloučit do jediného řetězce funkcí implode [5], kde první parametr je řetězec, kterým se mají jednotlivé buňky pole oddělovat a druhý parametr určuje pole, jehož prvky se mají sloučit.

Stejný postup se používá jak při odesílání XML dokumentu, tak při odesílání CNF souboru, který je nepovinný. Po načtení obou souborů do proměnných je nutné je zakódovat pomocí base64 a po té zavolat serverové funkci. Této předáme zakódované soubory XML a CNF a případně také pngdpi určující velikost PNG obrázku. V našem případě (tedy voláním funkce gantt2png) se vrátí klientovi soubor PNG.

```
$encodedString = base64_encode($string);
```

Kód 3.12) Zakódování stringu pomocí base64

```
$ganttReport = $soapclient->call('gantt2png',
$params = array('XML' => $XML, 'CNF' => $CNF, 'dpi' => $dpi),
$options);
```

Kód 3.13) Volání metody gantt ze serveru

Výstupem serveru je tedy soubor zakódovaný pomocí kódování base64 do jedné proměnné typu string, kterou jsem pojmenoval $ganttReport. Po ukončení běhu serverové funkce je třeba obdržený soubor uložit a případný CNF soubor zobrazit uživateli tak, aby jej bylo možné editovat. Přijaté soubory se rozkódují a uloží následujícím algoritmem:

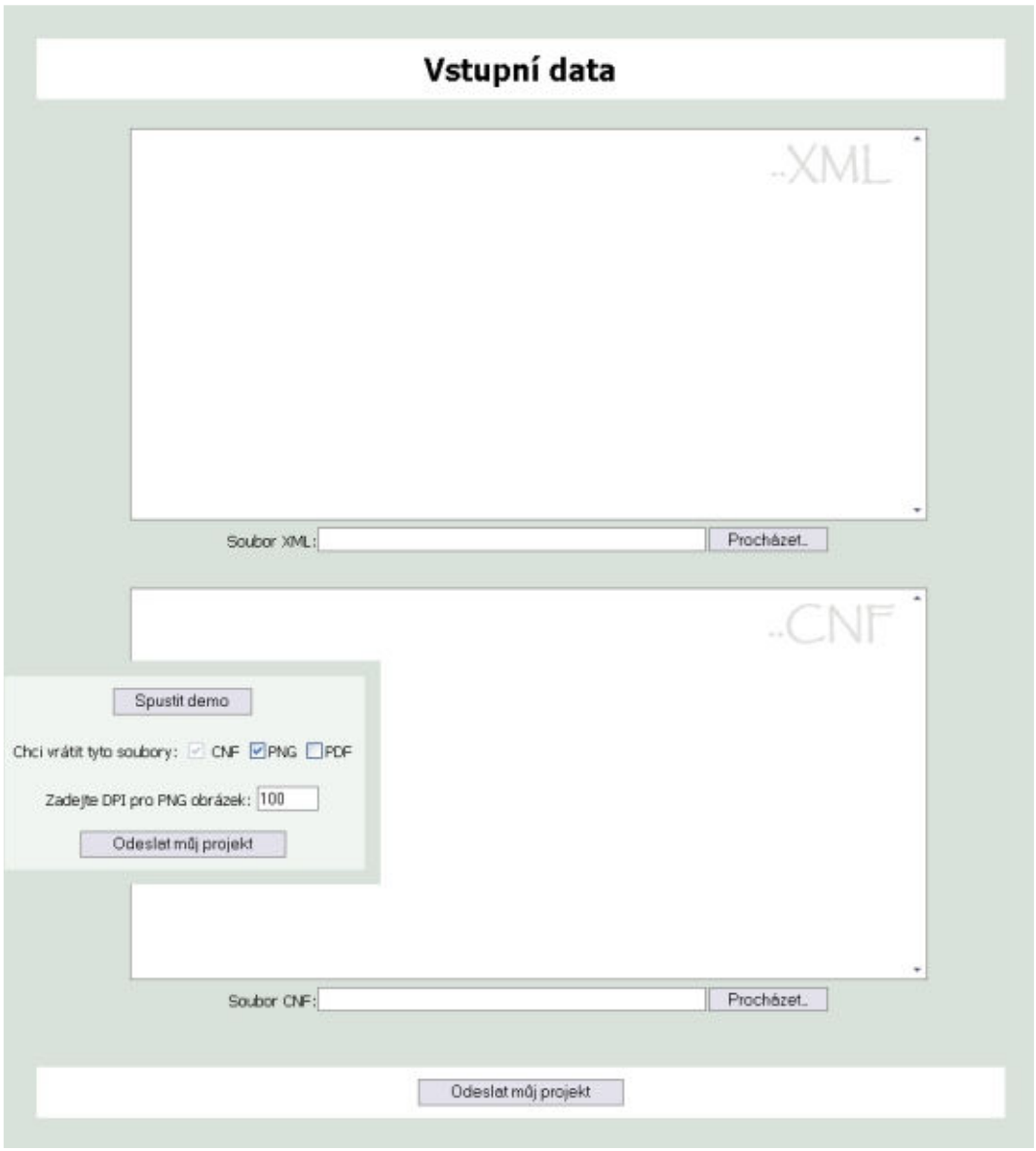
```
function decodeString2File($str,$filename)
{
        $fp = fopen($filename,'w');
        fwrite($fp,base64_decode($str));
        fclose($fp);
}
```

Kód 3.14) decodeString2File

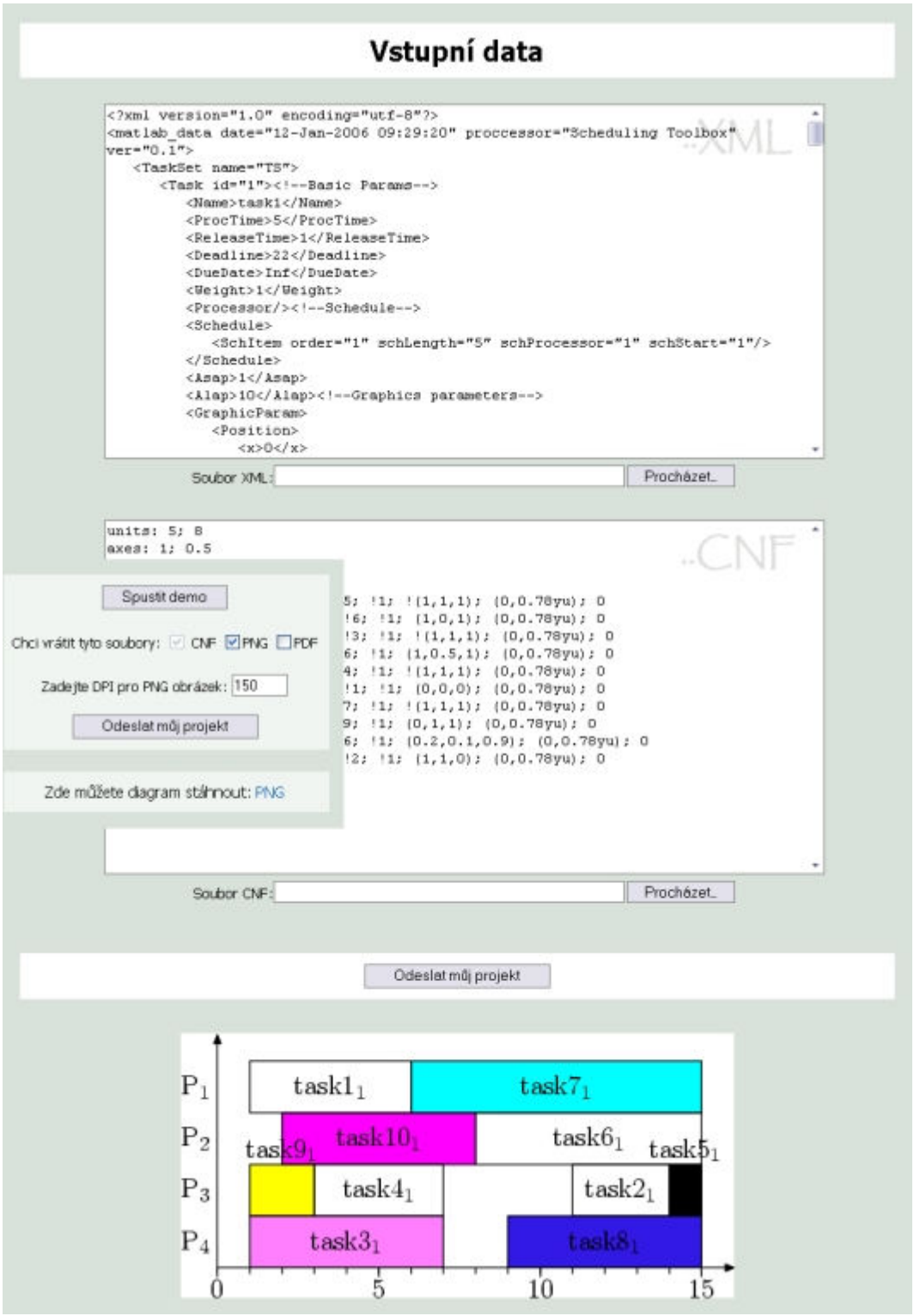
Do proměnné $fileName vložíme unikátní jméno vygenerované ihned po spuštění klientského skriptu (Kód 3.10), pod kterým soubor uložíme do zmíněné složky GanttOutput_PhpClient.

CNF soubor neukládáme, ale rovnou jej zobrazíme do formuláře uživateli, který jej může upravit a opětovným odesláním dat vygenerovat diagram pozměněný.

3.2.1.3 Screenshoty



Obrázek 3.3) Vzhled formuláře pro zadávání vstupních dat s ukázkou ovládacího panelu



Obrázek 3.4) Formulář s výsledným diagramem a ukázkou menu pro stažení

3.2.2 Klient - Matlab

3.2.2.1 Matlab a WebServices

Využívání WebServices je v Matlabu velmi dobře podporováno základní funkcí `createClassFromWSDL` [9]. Spouštíme ji s parametrem obsahujícím adresu WSDL dokumentu, pomocí nějž chceme vyhledat dostupné metody, jejich parametry a návratové hodnoty. Na základě tohoto dokumentu se v Matlabovském pracovním adresáři vytvoří složka (objekt), jejíž název začíná „@“, a který obsahuje pro každou dostupnou metodu jeden m-file. Za znakem „@“ je označení WebService odpovídající názvu serverového souboru s definicemi dostupných funkcí (v našem případě se serverový soubor jmenuje GanttServer.php, takže složka ponese jméno „@GanttServerService“, kde slovo „Service“ Matlab přidal jako označení WebServices)

Mimo zmíněné soubory složka obsahuje další m-file, tentokrát s názvem odpovídajícím názvu složky bez zavináče, tedy „GanttServerService.m“. Tento m-file je vlastně konstruktorem a obsahuje jen deklaraci a naplnění dvou proměnných. A to proměnné „endpoint“, která obsahuje adresu souboru „server.php“ a proměnná „wsdl“ s adresou WSDL dokumentu, tedy „server.php?wsdl“. Na tyto soubory se program odkazuje při odesílání dat serveru.

Díky tomu, že se ke každé funkci vytvoří m-file, je možné funkce volat téměř stejným způsobem jako funkce Matlabu. Jediný rozdíl je v tom, že prvním parametrem každé funkce, která využívá WebServices je objekt GanttServerService.

V každém z m-filů definujících dostupné funkce jsou uvedeny názvy a typy parametrů dané serverové funkce. Dále se v témže m-filu spouští funkce `createSoapMessage` [9], která z obdržených parametrů vygeneruje XML zprávu pro SOAP a ta je následně funkcí `callSoapService` [9] odeslána serveru. Po obdržení odpovědi se přijaté XML rozkóduje funkcí `parseSoapResponse` [9] tak, aby s výsledkem mohl Matlab pracovat, či jej zobrazit.

Výhodou v Matlabu je, že se vytvořená složka @GanttServerService nemaže a její obsah zůstává nezměněn. Není tedy nutné při každém spuštění WebServices využívat funkci createClassFromWSDL. Pokud se metody na serveru nemění, (ve smyslu změny parametrů, jmen atp.) tak stačí tuto funkci zavolat jednou, tedy poprvé. Toho jsem ve své práci využil a po ukončení vývoje potřebu volání funkce createClassFromWSDL odstranil. To se povedlo změnou m-filů obsahujících popis serverových metod tak, aby se nebylo nutné odkazovat na objekt GanttServerService. Adresu na „endpoint“ jsem zadal napevno a mohl tak zrušit volání funkce pro vytváření třídy. Pozůstatkem složky @GanttServerService tak je pouze jeden soubor pro každou poskytovanou metodu. Jde o metody xml2cnf, gantt2png, gantt2pdf.

3.2.2.2 Schéma funkce klientského skriptu

Matlabí klient funguje podobně jako klient v PHP. Opět jsou vstupními daty soubory XML a CNF, které je třeba zakódovat (pomocí base64), odeslat serveru a obdržet od něj výstupní data. Dále je možné si parametrem pngdpi zvolit velikost výstupního PNG obrázku.

Hlavním souborem je zde ganttlab.m, obsahující stejnojmennou metodu, která spouští pomocné funkce sloužící například ke kódování souborů, či uložení přijatých dat. Z metody ganttlab se také spouští serverové metody pro tvorbu diagramu. Ostatní soubory se jmenují stejně jako funkce, kterou nabízejí. Patří sem soubory decode a encode sloužící pro zakódování a rozkódování ochozích a příchozích souborů kódováním base64. Kódovat lze jen proměnnou typu string, proto je nutná funkce file2String a k ní inverzní string2File.

Pro zajištění unikátního jména, pod kterým se přijaté sobory uloží, slouží dvojice funkcí getPrefix a max [9]. První zmíněná funkce zjistí, jestli jméno, pod kterým se soubor snažíme uložit není již ve složce použito. Došlo by totiž k jeho přepsání, čemuž jsem chtěl zabránit. Pokud dané jméno již existuje, vrátí tato funkce číslo, které by bylo vhodným prefixem daného souboru. Toto se provede pro každý ukládaný soubor a funkcí Matlabu max se vybere nejvyšší prefix. To proto, aby vždy

série generovaných souborů měla zaručené stejné jméno. Níže uvádím kód zmíněné funkce getPrefix.

```
function [prefix] = getPrefix(filename)

prefix=1;
backupFileName = filename;

while ((exist(filename,'file') > 0))
   pre = num2str(prefix);
   filename=strcat(pre,bckupFileName);
   prefix=prefix+1;
end
prefix=prefix-1;
```

Kód 3.15) getPrefix

V metodě getPrefix je použita funkce Matlabu exist [9], která vrátí nulu, když soubor nenajde. Dále jsem pro sloučení dvou stringů využil funkci strcat [9].

Mezi další metody patří getCNF, getPNG a getPDF, které jen spustí příslušné serverové funkce s patřičnými parametry a vrátí jeden konkrétního soubor (CNF, PNG, PDF). Funkce showPNG zobrazí PNG obrázek a ganttlabDemo je metoda, která uživateli vytvoří demo projekt bez nutnosti zadávání jakýchkoli parametrů.

3.2.2.3 Popis implementace klienta

Po spuštění hlavní funkce ganttlab se převedou parametrem obdržené XML a CNF soubory na řetězce a to funkcí file2String, která načte soubor do proměnné pomocí Matlabovské funkce fread [9].

```
ganttlab(XmlFile,CnfFile,choice,dpi,name)
```

Kód 3.16) Schéma spuštění funkce ganttlab

Proměnná choice zde určuje jaké soubory budeme chtít vrátit. Funkce ganttlab podle tohoto parametru spustí odpovídající serverovou funkci a zpracuje data. Přehled hodnot choice uvádí následující tabulka:

Hodnotou proměnné choice	vyžádáme soubor:
1	CNF
2	PNG
3	PDF

```
function [str] = file2String(soubor)
fid = fopen(soubor,'r');
str = fread(fid,'*char')';
```

Kód 3.17) file2String

Prvním parametrem funkce fread je ukazatel na soubor k načtení. Druhý parametr značí, do jaké formy se má soubor načíst.

Výsledkem daného příkazu je načtení celého textového souboru do jednoho stringu. Tento algoritmus se aplikuje jak na XML, tak na CNF soubor. Pak je možné řetězec zakódovat funkcí encode a odeslat data serveru například spuštěním následující funkce:

```
getPNG(Xml,Cnf,dpi,pngName)
```

Kód 3.18) Příklad spouštění metody gantt2png z Matlabu pomocnou funkcí getPNG

Prvními dvěma parametry předáváme funkci proměnné obsahující XML a CNF soubor ve formě zakódovaného řetězce, dále proměnnou dpi, stejně jako v PHP klientovi, určujeme velikost PNG obrázku. Posledním parametrem je jméno, pod kterým se má soubor uložit. Toto jméno bylo odvozeno od posledního parametru funkce ganttlab (Kód 3.16) funkcí (Kód 3.15)

Ukládání souboru probíhá pomocí funkce string2File, kterou využívá funkce getPNG. Metodě string2File jako parametr předáme funkcí decode rozkódovaný řetězec a název cílového souboru.

```
string2File(decode(PNG),pngName);
```

Kód 3.19) Spouštění fce string2File

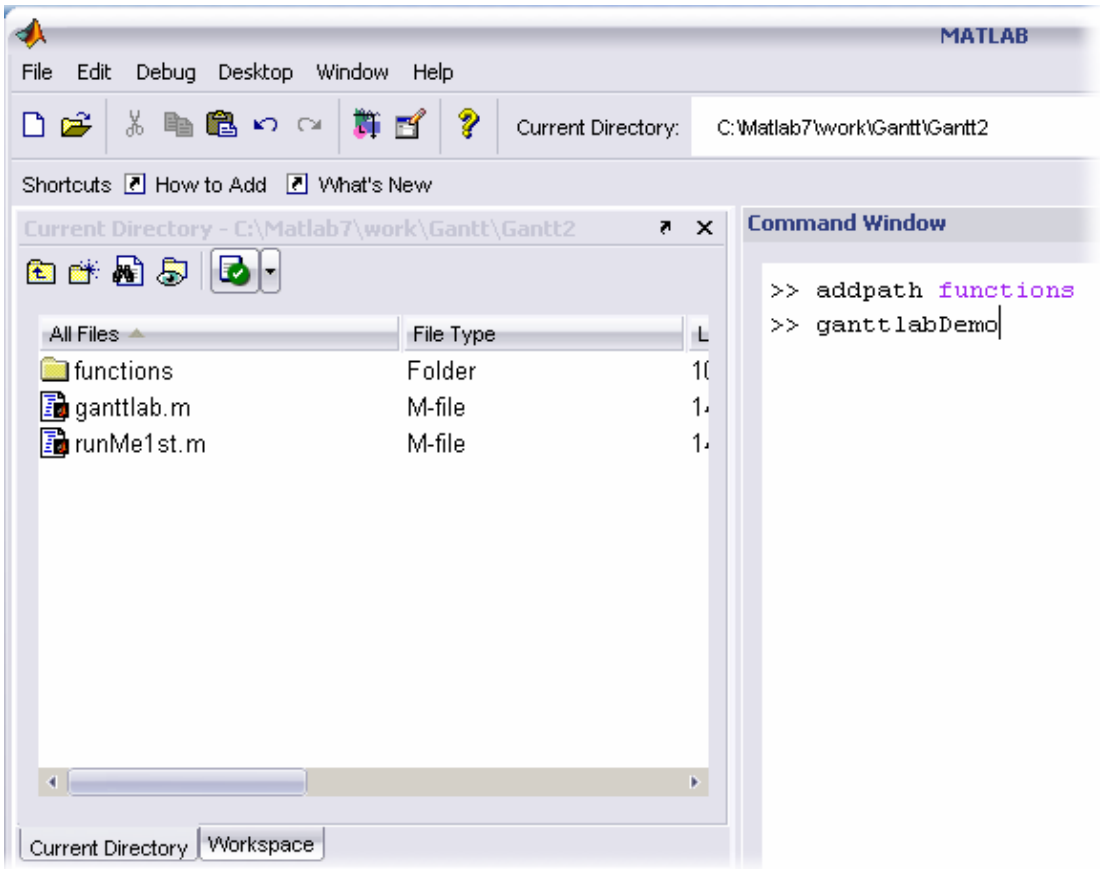
```
function string2File(string,filename)

fid = fopen(filename, 'w+');
dlmwrite(filename,string,'');
```

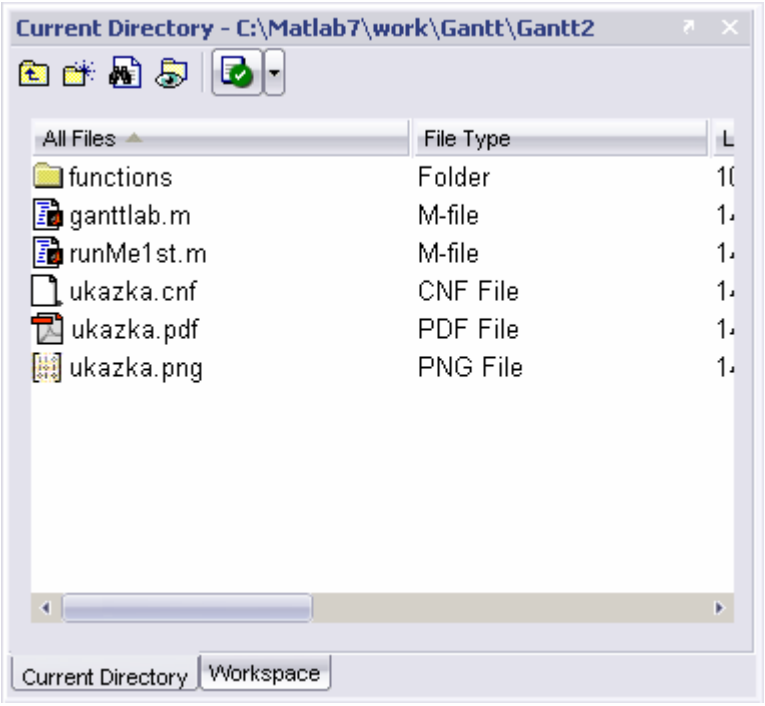
Kód 3.20) string2File

Po uložení souborů se uživateli zobrazí PNG obrázek některou Matlabovskou funkcí, například imshow, imview, imtool, imageview, nebo pomocí dvojice funkcí imread a image. Podle toho, kterou daná verze Matlabu disponuje, což ošetřuje moje funkce showPNG na základě zjišťování existence m-filu obsahující zobrazovací funkci.

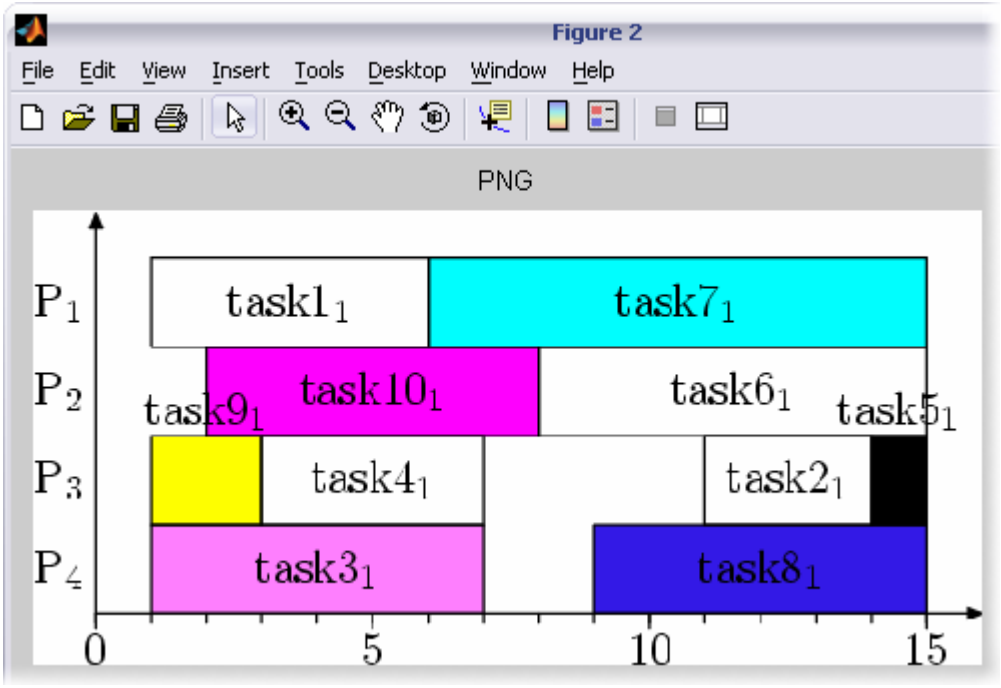
3.2.2.4 Screenshoty



Obrázek 3.5) Screenshot Matlab - spouštění



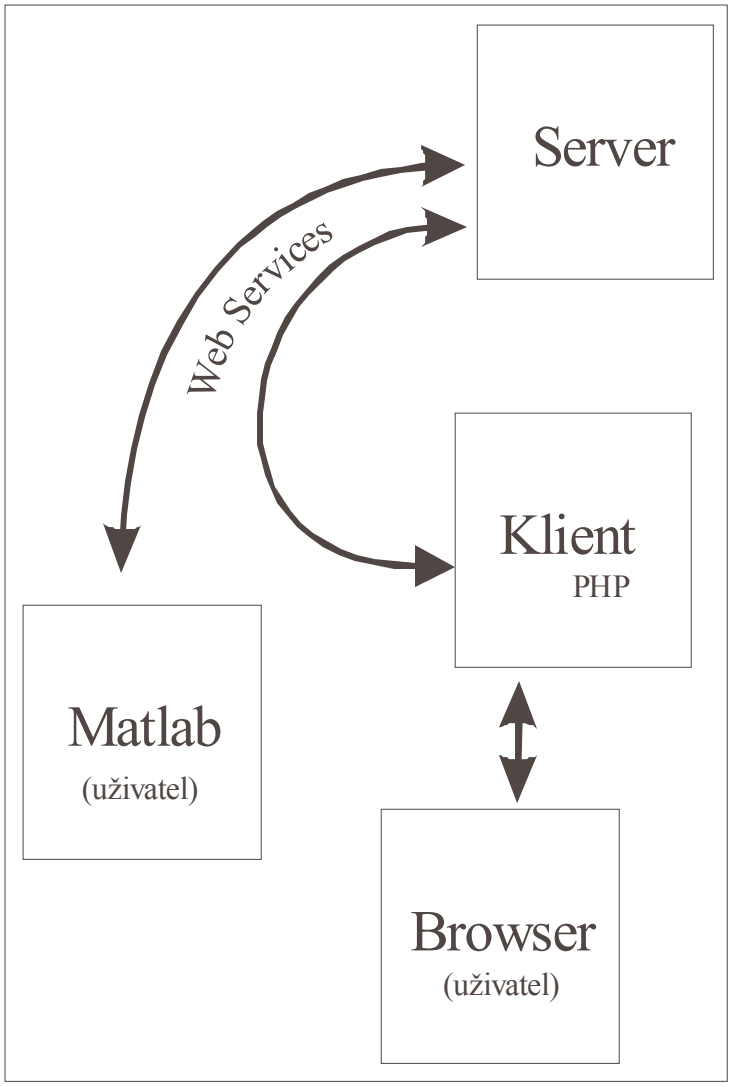
Obrázek 3.6) Screenshot Matlab – nové soubory



Obrázek 3.7) Screenshot Matlab – Zobrazení PNG obrázku

4 Výsledné schéma aplikace

Výsledkem mé práce jsou skripty komunikující dle schématu na Obrázku 4.1, kde blok „server“ zahrnuje soubory server.php a GanttServer.php. Pod blokem „klient“ se skrývá soubor index.php. Veškeré mé m-files pak spadají do bloku „Matlab“.



Obrázek 4.1) Výsledné schéma aplikace

5 Závěr a možnosti rozšíření do budoucna

Úkolem bylo napsat programy, které umožní tvorbu ganttových diagramů z webu a z Matlabu, což jsem splnil. Každý z klientů měl být schopen vyžádat si od serveru jen některé výstupní soubory a v případě PNG obrázku i jeho velikost. I toto mé programy umí. Drobný nedostatek by mohl být spatřován ve skutečnosti, že Matlabovský klient nemá GUI, na druhou stranu se však jednoduše ovládá a veškeré přijaté dokumenty jsou přehledně zobrazeny v pracovním adresáři, takže GUI není tolik důležité.

Do budoucna by bylo možné klienty rozšířit o editor konfiguračního souboru, který by uživateli zpříjemnil práci a zjednodušil tvorbu diagramů. Dalo by se též uvažovat o editoru XML souboru. Dle mých informací by program Gantt měl časem využívat XML dokument, který v sobě bude CNF soubor v nějaké formě obsahovat, čímž by se implementace klientů zjednodušila a editor by byl patrně nutný.

6 Odkazy

[1] Černý R. *Vzdálené generování ganttových diagramů* [online] 15.3.2006,
Poslední revize 10.7.2006 [citováno 15.června 2006]
<http://rtime.felk.cvut.cz/~cernyr1/gantt/index.php>.

[2] Černý R. V*zdálené generování ganttových diagramů - Server.php* [online]] 15.3.2006,
Poslední revize 10.7.2006 [citováno 15.června 2006]
<http://rtime.felk.cvut.cz/~cernyr1/gantt/index.php>.

[3] Černý R. *Vzdálené generování ganttových diagramů - GanttServer.php* [online] 15.3.2006,
Poslední revize 10.7.2006 [citováno 15.června 2006]
<http://rtime.felk.cvut.cz/~cernyr1/gantt/GanttServer.php>.

[4] ImageMagick. *ImageMagick: Command-line Tools: Convert* [online],
[citováno 15.června 2006]
<http://www.imagemagick.com/www/convert.html>

[5] The PHP Group. *PHP: Hypertext Preprocessor* [online], [citováno 15.června 2006]
<http://cz.php.net/>

[6] http://en.wikipedia.org/wiki/Base64

[7] The PHP Group. *PEAR :: Package :: SOAP* [online], [citováno 15.června 2006]
<http://pear.php.net/package/SOAP>

[8] Karolík A. *PSGANTT – RTIME* [online], [citováno 15.června 2006]
<http://rtime.felk.cvut.cz/wiki/index.php/PSGANTT>

[9] Mathworks. *The MathWorks - Online Documentation, R2006a* [online],[citováno 15.června 2006]
<http://www.mathworks.com/access/helpdesk/help/helpdesk.html>

[10] Kutil M. *TORSCHE Scheduling Toolbox for Matlab* [online],
[citováno 15.června 2006]
<http://rtime.felk.cvut.cz/scheduling-toolbox>

[11] Kutil M. *TORSCHE Scheduling Toolbox for Matlab - Manual - Introduction* [online],
[citováno 15.června 2006]
<http://rtime.felk.cvut.cz/scheduling-toolbox/manual/introduction.php>

[12] W3C. *World Wide Web Consortium* [online],[citováno 15.června 2006]
<http://www.w3.org>

[13] Kosek J. *XML schéma* [online] 2003,
Poslední revize 15.8.2005 [citováno 15.června 2006]
<http://www.kosek.cz/xml/schema>

[14] Pikner T. *Standardy rodiny XML* [online] 2001, [citováno 15.června 2006]
<http://www.fi.muni.cz/~tomp/slides/pb138/1std-terminology/standards/toc.html>

[15] Kuba M. *WebServices* [online], [citováno 15.června 2006]
<http://www.ics.muni.cz/bulletin/issues/vol13num03/kuba2/kuba2.html>