ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

DIPLOMOVÁ PRÁCE

Editor pro stochastické Petriho sítě na WWW

Praha, 2006 Tomáš Ivánek

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky *Školní rok:* 2004/2005

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Tomáš Ivánek

Obor: Technická kybernetika

Název tématu: Editor pro Stochastické Petriho sítě na WWW

Zásady pro vypracování:

1. Vytvořte skripty pro parsing a vytváření SPN modelu v XML.
2. Zvolte jednu z metod řešení dodaných vedoucím práce a ověřte její správnost popř. navrhněte vlastní strategii.
3. Implementujte knihovnu v PHP pro interaktivní práci s grafickými objekty na www.
4. Implementujte obousměrné programátorské rozhraní mezi prohlížečem a serverem (kvůli návaznosti na další fáze projektu musí rozhraní umět reagovat i na události vzniklé na straně serveru).
5. Vytvořte grafický editor SPN s použitím knihovny v PHP.
6. Volitelně: Vytvořte skript v PHP, který provede analýzu některé (některých) vlastnosti Petriho modelu a jeho integraci do editovacího prostředí.

Seznam odborné literatury:

[1] Kosek Jiří: *PHP tvorba interaktivních internetových aplikací*. Grada 2000

[2] Flanagan David: *JavaScript Kompletní průvodce*. Computer Press 2002, 2.aktualizované vydání

Vedoucí diplomové práce: Ing. Josef Čapek, Ph.D.

Termín zadání diplomové práce: zimní semestr 2004/2005

Termín odevzdání diplomové práce: leden 2006

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

v zast.

prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 07.03.2005

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ……. …….

Tomáš Ivánek

Poděkování:

Chtěl bych zde poděkovat rodičům za jejich podporu při studiu. Dále pak Ing. Josefu Čapkovi Ph.D., bez jehož rad a připomínek by tato práce sotva vznikla.

ANOTACE

NÁZEV: EDITOR PRO STOCHASTICKÉ PETRIHO SÍTĚ NA WWW.

Cílem této diplomové práce je navrhnout a implementovat grafový editor jako tenkého klienta ve webovém prohlížeči. Editor je implementován pomocí jazyka HTML, JavaScript (ECMAScript) a PHP. Je navržen jako nenáročný a interaktivní grafový editor s propojením na databázový server s uživatelskými definicemi grafů a vytvořenými grafovými modely.

Editor pracuje na webovém serveru, na straně klienta a klient je propojen s databází pro ukládání dat.

ABSTRACT

TITLE: STOCHASTIC PETRI NET EDITOR ON WWW.

The goal of this diploma work is to design and implement graph editor, like thin client in web browser. Editor is implemented in HTML, JavaScript (ECMAScript) and in PHP language. Is designed like modest and interactive editor with connection on database server with user definitions and created graph models.

Editor works on web server, on client side and client is connected to database for saving data.

OBSAH

1 Úvod ... 11
1.1 Technické požadavky ... 12

2 Použité technologie ... 13
2.1 XML (eXtensible Markup Language) ... 13
2.1.1 Úvod ... 13
2.1.2 Vlastnosti XML ... 13
2.2 Relax NG (REgular LAnguage for XML, New Generation) ... 15
2.2.1 Úvod ... 15
2.2.2 Struktura ... 15
2.3 HTML ... 16
2.4 CSS ... 16
2.5 W3C DOM ... 17
2.6 Javascript (ECMAScript) ... 18
2.7 PHP ... 18
2.8 AJAX (Asynchronous JavaScript and XML) ... 19
2.9 Protokol HTTP ... 19
2.9.1 Činnost protokolu ... 20
2.9.2 POST a GET metody ... 20
2.10 Graf (teorie grafů) ... 20
2.11 Petriho sítě ... 21
2.11.1 Základní názvosloví ... 21
2.11.2 Petriho sítě s časovanými přechody ... 22
2.11.3 Stochastické Petriho sítě ... 22
2.11.4 Stochastické časované Petriho sítě ... 23
2.12 PNML ... 24
2.13 StochPNTD ... 24
2.13.1 Dodefinované vlastnosti ... 24

3 Architektura obecného editoru grafů v JS ... 27
3.1 Kreslící plátno ... 27
3.2 Kreslení ve webovém prohlížeči ... 28
3.2.1 Statické metody kreslení ... 28
3.2.2 Dynamické metody kreslení ... 29
3.2.3 Metodika měření zkreslení ... 31
3.2.4 Nakreslené předgenerované úsečky ... 33
3.3 Objekt vrchol ... 35
3.3.1 Algoritmus vytváření objektu ... 35
3.3.2 Vnitřní datová struktura objektu vrchol ... 37
3.3.3 Příklad použití objektu v JS ... 38
3.4 Objekt hrana ... 39
3.4.1 Obecný popis ... 39
3.4.2 Vnitřní datová struktura objektu hrana ... 42
3.4.3 Příklad použití objektu v JS ... 42
3.5 Popisek ... 43

3.6 PHP soubory pro generování grafických objektů ... 44
3.6.1 Generování vrcholů ... 44
3.6.2 Generování textu ... 46
3.6.3 Generování hran ... 46
3.7 Grafické jádro editoru ... 48
3.7.1 Datová reprezentace grafu: ... 48
3.7.2 Funkce pro výpočet průsečíku kružnice–úsečka ... 50
3.7.3 Funkce pro výpočet průsečíku obdélník–úsečka ... 51
3.7.4 Mazání objektů ... 53
3.7.5 Změna vlastností vrcholu ... 53
AJAX – Asynchronní komunikace se serverem ... 54
3.7.6 Použití AJAX technologie ... 55
3.7.7 Technologie řešení ... 56
3.7.8 Schéma SQL databáze pro ukládání dat za běhu ... 56
3.7.9 Načítání grafu ... 58
3.8 Generování XML souboru ... 59
3.8.1 Struktura StochPNTD ... 60
3.9 Podpůrné definice ... 62
3.9.1 Úvod ... 62
3.9.2 Technologie parsování definičního souboru ... 62
3.9.3 Propojení s klientem ... 62

4 Specializace editoru ... 64
4.1 Organizační diagram ... 64
4.2 Petriho sítě ... 64

5 Závěr ... 65

6 Přílohy ... 66
6.1 Příloha A – uživatelská příručka k editoru ... 66
6.1.1 Úvodní obrazovky ... 66
6.1.2 Hlavní obrazovka a použití jednotlivých nástrojů ... 67
6.2 Příloha B – obsah přiloženého CD ... 69

7 Literatura a reference ... 70

SEZNAM OBRÁZKŮ:

Obrázek 1 – Jednoduchá Petriho síť 21
Obrázek 2 – Hustota pravděpodobnosti *f(x)* rovnoměrného rozdělení 23
Obrázek 3 – Okno prohlížeče Internet Explorer a kreslící plocha 27
Obrázek 4 – Horizontální a vertikální posuvníky v prohlížeči 28
Obrázek 5 – Kreslení pomocí objektů DIV 29
Obrázek 6 – Parametry hrany při vykreslení 30
Obrázek 7 – Srovnání aliasing a antialiasingu nakreslené hrany (zvětšeno) 31
Obrázek 8 – Graf závislosti zkreslení ve srovnání s ideální hranou 33
Obrázek 9 – Kvadranty pro kreslení hrany 34
Obrázek 10 – DOM struktura dokumentu HTML v prohlížeči 35
Obrázek 11 – Vývojový diagram vytváření DIVu 37
Obrázek 12 – Vývojový diagram změny hrany 39
Obrázek 13 – Schéma kreslení hrany při různých délkách a koncových souřadnicích 40
Obrázek 14 – Ukázka nakreslených hran 41
Obrázek 15 – Nakreslená hrana s koncovou šipkou 41
Obrázek 16 – Vlevo šedivé pozadí vrchního kruhového vrcholu,objekt vpravo má transparentní pozadí 44
Obrázek 17 – Příklady nakreslených objektů s různými grafickými parametry 45
Obrázek 18 – Struktura PHP souboru pro generování vrcholů 45
Obrázek 19 – Hrana před výměnou obrázku JavaScriptem 47
Obrázek 20 – Hrana po výměně obrázku JavasSriptem 47
Obrázek 21 – Rozvržení objektů na kreslící ploše 48
Obrázek 22 – Struktura objektů typu vrchol 49
Obrázek 23 – Struktura objektů typu hrana 49
Obrázek 24 – Obrázek ilustruje výsledný graf (Model 2) po pohybu vrcholů v souřadnicích [x,y] 50
Obrázek 25 – Výpočet průsečíku kružnice s úsečkou 50
Obrázek 26 – Počítání průsečíku obdélníku strana (a>b) a úsečky 51
Obrázek 27 – Počítání průsečíku úsečky se čtvercem 52
Obrázek 28 – Počítání obdélníku strana (a<b) a úsečky 52
Obrázek 29 – Vlastnosti kreslených objektů 53
Obrázek 30 – Obecné schéma aplikace využívající AJAX technologii, převzato a přeloženo z [42] 54
Obrázek 31 – Technologie zpráv mezi klientem a serverem 56
Obrázek 32 – Schéma databáze 57
Obrázek 33 – Vygenerování a připojení JS souboru pomocí technologie SAJAX 59
Obrázek 34 – Popis algoritmu parsování definičního souboru 62
Obrázek 35 – Organizační diagram turnaje 64
Obrázek 36 – Schéma Petriho sítě 64
Obrázek 37 – Úvodní obrazovka 66
Obrázek 38 – Výběr možností editoru 66
Obrázek 39 – Hlavní obrazovka editoru 67

SEZNAM TABULEK:

Tabulka 1 – Přípustné hodnoty atributu type a jejich význam 25
Tabulka 2 – Podelementy elementu <time> v závislosti na hodnotě atributu type. 26
Tabulka 3 – Parametry objektu vrchol 36
Tabulka 4 – Vlastnosti objektu vrchol 37
Tabulka 5 – Vlastnosti objektu vrchol (pokračování) 38
Tabulka 6 – Vlastnosti objektu hrana 42

SEZNAM POUŽITÝCH ZKRATEK A TERMINOLOGIE

MS IE: Microsoft Internet Explorer.

Internetový prohlížeč: Prohlížeč s grafickým renderovacím jádrem. (IE, FireFox, Opera, Konqueror a další) podporující standard W3C HTML 4.0.

Pixel: Je nejmenší jednotka digitální rastrové (bitmapové) grafiky.

JS: JavaScript.

Vrchol: Graf je uspořádaná dvojice (V, E), prvky množiny V se nazývají vrcholy grafu.

Hrana: Graf je uspořádaná dvojice (V, E), prvky množiny E se nazývají hrany.

Popisek: Text identifikující objekt.

DOM: Objektový model dokumentu.

DIV: Typ oddílu, zahrnuje v sobě libovolně velkou oblast textu včetně nadpisů, obrázků a tabulek.

HTML: Značkovací jazyk pro hypertext.

XML: Rozšiřitelný značkovací jazyk.

AJAX: (Asynchronous JavaScript and XML) je technologie určená pro vývoj interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání.

PHP: Programovací jazyk určený především pro programování dynamických internetových stránek.

Pozn.: Podrobnější popis použitých technologií je uveden v kapitole 2.

1 Úvod

S vývojem internetu dochází k rozvoji webových služeb. Současným trendem jsou webové aplikace, které se použitými prvky interaktivity svého chování přibližují možnostem desktopových aplikací. K hlavním výhodám webových aplikaci patří platformní nezávislost, snadná a bezproblémová šířitelnost a velká operativnost při jejich nasazení. Platformní nezávislost přináší možnost nasazení pod různými operačními systémy. Bezproblémová šířitelnost znamená, že naprostá většina uživatelů z dané cílové skupiny bude schopna velmi snadno aplikaci spustit a používat. Současný stav řešení otázek důvěry a bezpečnosti v IT s sebou přináší omezení při šíření desktopových aplikací. Např. je známo, že mnoho uživatelů z důvodu obavy z napadení virem vůbec nestahuje desktopové aplikace vystavené na internetu.

Interaktivní internetové aplikace se mohou vytvářet např. pomocí Java apletů[1], ActiveX[2] , Flash[3] nebo JavaScriptu a DHTML[4]. Aplety řeší platformní nezávislost (Windows, Linux, Mac OS ..), jejich rychlost je teoreticky porovnatelná s desktopovou aplikací a může se jednat o aplikace pracující v reálném čase. Mezi jejich nevýhodami je pak nutnost mít nainstalovaný Java plugin do prohlížeče (to některých nemusí být k dispozici) a správnou verzi JVM[5] . Aplety se spouští velmi dlouho (než se spustí napoprvé prostředí JVM), jejich velikost je většinou velká a aplety se musí celé stáhnout na klientskou stranu a to většinou zabere nějaký čas. Pokud je aplet nedůvěryhodný, má omezený přístup na uživatelský systém (přímý přístup na klientský disk nebo schránku). Mezi neopominutelné nevýhody patří velká paměťová náročnost. Uživatelé mnohdy právě z důvodů bezpečnosti a nerozhodnosti při odlišení bezpečného apletu od nebezpečné aplikace aplety nestahují. Navíc může být na některých sítích stahování apletů administrativně zakázáno. Další možností je použít technologii ActiveX, která je podobná apletům, ale přináší bezpečností problémy (z tohoto důvodu je ve většině firmách z bezpečnostních důvodů tato technologie zakázána). ActiveX jsou použitelné jen v Microsoft Internet Exploreru a tak tuto technologii nelze použít na jiných operačních systémech (Linux, Solaris, Mac OS..).

Požadavek na webové aplikace je, aby aplikace byly přístupné všude a měly malé nároky na softwarové vybavení. Proto technologie DHTML a Javascript nabízejí výhodu přenositelnosti a malé datové náročnosti. Rovněž hardwarové nároky na klientský počítač a celková velikost aplikace je mnohem nižší než u desktop aplikace. Webové aplikace mají výhodu platformní nezávislosti, globálního obsahu – aplikaci mohou využívat uživatelé z celého světa a není potřeba instalovat klientskou část aplikace, kterou zajistí standardní prohlížeč. Znamená to, že

---

[1] Aplet je program, napsaný v Javě, který je začleněn do HTML stránky.
[2] ActiveX technologie vyvinutá společností Microsoft pro sdílení informací mezi různými aplikacemi. Komponenta ActiveX může být například automaticky zavedena a spuštěna prohlížečem webových stránek.
[3] Flash je grafický vektorový program používaný pro tvorbu (převážně internetových) interaktivních animací, prezentací a her.
[4] Dynamické HTML je kód vzniklý sloučením HTML a CSS.
[5] JVM (Java Virtual Machine). Virtuální stoj, na kterém může být spuštěn kód kompilovaný Javou.

uživatel může k aplikaci přistupovat z jakéhokoliv počítače s grafickým webovým prohlížečem a nejsou kladeny velké nároky na softwarové vybavení.

Mezi nejvyužívanější aplikace patří přístup do schránky elektronické pošty (např. [31]), klient pro posílání a příjem krátkých textových zpráv (IM[6]) [34], plánovací kalendář [29], portály pro sdílení fotografií [35], kancelářské programy [32] a [33] nebo internetové obchody.

Internet byl koncipován jako nestavový protokol. Z toho vyplývají nevýhody, které lze použitím vhodných technologií částečně odstranit (zejména technologií AJAX a použitím databáze). Na internetu jsou k dispozici aplikace pro práci s Petriho sítěmi, jsou to ale jen neinteraktivní aplikace určené pro vykreslování nebo převod PNML[7] grafu. Lze je vyzkoušet na [43] a [44].

Aplikace naprogramovaná v rámci této diplomové práci dovoluje interaktivní kreslení a úpravu grafu. Aplikace je koncipována jako klient–server aplikace. V editoru lze otevřít, navrhnout obecný graf, editovat ho, uložit ho na server a poté pokračovat v práci, nebo exportovat ve formátu StochPNTD[8] (vycházející z PNML, rozšířeného o stochastické a další grafické vlastnosti).

Na serveru se nachází databáze využívaná při kreslení a načítání ukládaných grafů. Databáze je využívaná při kreslení a odstraňuje nevýhody vzniklé použitím generování dynamického obsahu stránky.

1.1 Technické požadavky

Pro běh editoru je nutné připojení k internetu a prohlížeč. Protože se jedná o interaktivní grafový editor je třeba použít prohlížeč s grafickým jádrem.

Mezi podporované prohlížeče patří Microsoft Internet Explorer verze 5.5 a vyšší , Mozilla Firefox 1.5[9], Opera. 8.5. Prohlížeč musí mít zapnutou podporu jazyka JavaScript.

[6] IM (Internet Messaging) – aplikace umožňující textovou komunikaci mezi uživateli.
[7] PNML je jazyk popisující Petriho sítě, založený na technologii XML
[8] StochPNTD vznikl rozšířením PNML.
[9] Z důvodu nestandardní implementaci JavaScriptu nemusí všechny vlastnosti editoru fungovat.

2 Použité technologie

2.1 XML (eXtensible Markup Language)

2.1.1 Úvod

Zkratka *XML* znamená eXtensible Markup Language, česky rozšiřitelný značkovací jazyk. Byl vyvinut konsorciem W3C[10] (World Wide Web Consortium) jako pokračování jazyka *HTML* a SGML[11].

Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury *XML*.

Původní jazyk pro publikování *HTML* již přestal vyhovovat především pro svou složitost, která vznikla jeho postupným (a svévolným) rozšiřováním. Jazyk *XML* nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější, než *HTML*.

2.1.2 Vlastnosti XML

Standardní formát pro výměnu informací

V dnešní době již není vhodné zasílat dokumenty ve tvaru, který vyžaduje speciální software některé firmy, jako je např. formát DOC, XLS nebo T602.[12] Je používána celá řada operačních a informačních systémů a není záruka, že každý uživatel vlastní příslušný software.

Je tedy potřeba používat nějaký jednoduchý otevřený formát, který není úzce svázán s platformou nebo proprietární technologií. Tím může být právě *XML*, který je založen na jednoduchém textu a je zpracovatelný (v případě potřeby) libovolným textovým editorem[13].

Specifikace *XML* konsorcia W3C je zdarma přístupná všem. Každý tak může bez problémů do svých aplikací implementovat podporu *XML*. To je velký rozdíl proti firemním formátům, k nimž není k dispozici žádná veřejná dokumentace a navíc se jedná v porovnání s *XML* o velice složité, často binární formáty.

Mezinárodní podpora

XML bylo navrhováno i s podporou jiných jazyků než je angličtina. Jako znaková sada se implicitně používá IEC/ISO 10646 (také Unicode). V *XML* proto můžeme vytvářet

---

[10] W3C – World Wide Web Consortium (W3C) je mezinárodní konsorcium jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web.
[11] SGML (Standard Generalized Markup Language) je standardní jazyk určený k formálnímu popisu struktury dokumentů. Je definován v normě IEC/ISO 8879 z roku 1986.
[12] Formáty dokumentů používané kancelářskými programy.
[13] Textovým editorem se myslí editor zvládající kódování utf–8.

dokumenty, které obsahují texty v mnoha jazycích najednou – můžeme přepínat mezi různými jazyky v jednom dokumentu. Současně je přípustné i jiné libovolné kódování (např. windows–1250, iso–8859–2), musí však být v každém dokumentu přesně určeno. Odpadají tak problémy s konverzí z jednoho kódování do druhého.

Vysoký informační obsah

Pomocí *XML* značek (tagů) vyznačujeme v dokumentu význam jednotlivých částí textu. Dokumenty tak obsahují více informací, než kdyby se používalo značkovaní zaměřené na prezentaci (vzhled) – definice písma, odsazení a podobně. *XML* dokumenty jsou informačně bohatší. To lze samozřejmě s výhodou využít v mnoha oblastech. Největší přínos bude samozřejmě pro prohledávání, kdy můžeme určit i jaký význam má mít hledaný text.

Snadná konverze do jiných formátů

Při používání *XML* dokumentu potřebujeme také dokument zobrazit. *XML* samo o sobě žádné prostředky pro definici vzhledu nenabízí. Existuje ale několik stylových jazyků, které umožňují definovat, jak se mají jednotlivé elementy zobrazit. Souboru pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu, se říká styl.

Jeden vytvořený styl můžeme aplikovat na mnoho dokumentů stejného typu, stejně tak můžeme na jeden dokument aplikovat několik různých stylů. Výsledkem může být např. postscriptový soubor, *HTML* kód nebo *XML* s obsahem původního dokumentu.

Stylových jazyků existuje dnes několik. Mezi nejznámější patří asi kaskádové styly (CSS). Ty lze použít pouze pro jednoduché formátování, které dobře poslouží pro zobrazení dokumentu na obrazovce. Dalším jazykem je XSL[14] (eXtensible Stylesheet Language). Ten umožňuje dokument různě upravovat a transformovat – vybírat části dokumentu nebo generovat obsahy a rejstříky.

Technologii CSS se věnuji v kapitole 2.4.

Automatická kontrola struktury dokumentu

XML neobsahuje předdefinované značky (tagy), je třeba definovat vlastní značky, které budeme používat. Tyto značky je možné definovat v souboru DTD (Document Type Definition). Potom je možné automaticky kontrolovat, zda vytvářený *XML* dokument odpovídá této definici. Program, který tyto kontroly provádí, se nazývá parser. Při vývoji aplikací můžeme parser použít a ten za nás detekuje většinu chyb v datech.

DTD není jediný definiční jazyk pro *XML*. Neobsahuje možnost kontrolovat typy dat (čísla, měnové údaje, informace o datu a čase). To je vlastnost, která chybí při zpracování dat databázového charakteru. V současné době se pod názvem „*XML* schémata“ pracuje na půdě konsorcia W3C na vytvoření jednotného standardu, který tyto kontroly umožní.

[14] XSL je skupina technologii zahrnující jazyky XSLT a XPath a XSLFO.

Pro různé standardní aplikace byla postupně vytvořena schémata, která definují značky (názvy elementů) pro konkrétní typy dokumentů. Příkladem může být DocBook, který definuje struktury pro vytváření knih, článků, vědeckých publikací a podobně. Výhodou takových aplikací je, že současně s definičními soubory DTD je dodávána sada stylů (XSL souborů) pro následné zpracování a přípravu pro tisk nebo pro převod do jiných standardních tvarů (PostScript, *HTML* atd.).

Další vlastností *XML* je, že v jednom dokumentu můžeme používat najednou nezávisle na sobě několik druhů značkovaní pomocí jmenných prostorů (namespaces). To umožňuje kombinovat v jednom dokumentu několik různých definic ve formě DTD nebo schémat bez konfliktů v pojmenování elementů.

2.2 Relax NG (REgular LAnguage for XML, New Generation)

2.2.1 Úvod

Přímo specifikace jazyka *XML* popisuje DTD (Document Type Definition), což je jazyk pro popis schématu dokumentu pocházející ještě z jazyka SGML. DTD díky tomu mají jednu velkou výhodu – podporuje je velké množství aplikací. To je však současně jejich jedinou výhodou.

S rozšiřováním *XML* začínaly být nedostatky DTD stále patrnější. Za největší nedostatek můžeme označit nulovou podporu pro jmenné prostory. Jmenné prostory jsou mechanismus, který umožňuje v jednom dokumentu *XML* kombinovat více sad značek.

Jako reakce na tento stav vznikl další schémový[15] jazyk Relax NG. Relax NG je nový a elegantní jazyk pro popis schématu *XML* souborů, standardizován ISO (International Standard ISO/IEC 19757–2).

Relax NG schéma je jednoduché, obsahuje syntaxi *XML*, s jeho pomocí lze popsat jakákoliv strukturovaná data a je zaměřeno na lepší validaci *XML* dokumentů.

2.2.2 Struktura

Strukturu dokumentu můžeme zapisovat dvojím způsobem.

Syntaxe založená na XML

Umožňuje snadné zpracování pomocí dalších nástrojů (např. generování dokumentace pomocí XSLT). Ruční zápis je zbytečně zdlouhavý.

Typická koncovka: rng.

[15] Jazyk pro popisující syntaktická pravidla dokumentu.

Kompaktní syntaxe

Úsporná textově orientovaná syntaxe umožňuje velmi rychlé ruční psaní. Výsledné schéma je velmi přehledné, protože je krátké.

Typická koncovka: rnc.

Kompaktní i *XML* syntaxe mají ekvivalentní vyjadřovací schopnosti.

Navzájem jsou kompatibilní a můžeme je převádět pomocí programu Trang [28].

2.3 HTML

HTML - zkratka z anglického HyperText Markup Language, značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu.

Jazyk je podmnožinou dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj *HTML* byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka.

Podrobnější specifikace jazyka HTML lze najít v [41].

2.4 CSS

CSS je zkratka pro anglický název Cascading Style Sheets, česky tabulky kaskádových stylů. Je to jazyk pro popis způsobu zobrazení stránek napsaných v jazycích *HTML*, *XHTML* nebo *XML*.

Jazyk byl navržen standardizační organizací W3C. Byly vydány zatím dvě verze specifikace CSS1 a CSS2. Připravovaná nová verze CSS3 se očekává v roce 2008.

Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk *HTML*, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze *HTML* obsahují celou řadu elementů, které nepopisují jen obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

Používání kaskádových stylů ve srovnání se samotným *HTML* v praxi přináší výhody:

- širší formátovací možnosti
- snadná tvorba a údržba konzistentního stylu
- oddělení struktury a stylu
- dynamická práce se styly
- formátování *XML* dokumentů
- větší kompatibilita alternativních webových prohlížečů
- kratší doba načítání stránky

Stylový předpis se skládá z posloupnosti pravidel. Každé pravidlo určuje vzhled některého elementu dokumentu, nebo skupiny elementů.

CSS dovolují různé formy zápisu:

Přímý zápis

Do zdroje *HTML* souboru se napíše tato deklarace odstavce:

```
<p style="color: red">Tento odstavec bude červený.</p>
```

Vysvětlení: <p> je značka vymezující odstavec; z anglického paragraph. Atribut "style" je obecný atribut použitelný u elementů dovolující formátování. *Color* znamená barva a *red* je červená.

Stylopis

Do hlavičky dokumentu se napíše stylopis uzavřený mezi tagy <style></style>:

```
<style>
p    {color: red}
</style>
```

a do těla stránky se mohou psát odstavce:

```
<p>Tento odstavec bude červený. </p>
<p>Tento mimochodem také, protože červené budou všechny.</p>
```

Externí CSS soubor

Vytvoří se soubor, který se pojmenuje třeba *styly.css*. V něm bude pouze tento text:

```
p    {color: red}
```

Do hlavičky *HTML* dokumentu, který chci stylem ovlivnit, musím napsat odkaz na tento soubor:

```
<link rel="stylesheet" type="text/css" href="styly.css">
```

V těle dokumentu pak budou opět všechny odstavce červené.

2.5 W3C DOM

DOM (akronym anglického Document Object Model – objektový model dokumentu) je objektově orientovaná reprezentace *XML* nebo *HTML* dokumentu. *DOM* je API umožňující přístup či modifikaci obsahu, struktury, nebo stylu dokumentu, či jeho částí.

Původně měl každý webový prohlížeč své vlastní specifické rozhraní k manipulaci s *HTML* elementy pomocí JavaScriptu. Vzájemná nekompatibilita těchto rozhraní však přivedla konsorcium W3C k myšlence standardizace, a tak vznikl *W3C Document Object Model* (zkráceně *W3C DOM*). Tato specifikace je platformě a jazykově nezávislá. Předchozí specifická rozhraní byla nazvána *Intermediate DOM* (anglicky *přechodný DOM*).

DOM umožňuje přístup k dokumentu jako ke stromu. Jednotlivé větve tvoří elementy, atributy, text, komentáře. Tato technologie, nazývaná grove (Graph Representation Of property ValuEs), vyžaduje nahrání celého parsovaného[16] dokumentu do paměti, z čehož plyne, že optimální použití technologie *DOM* je tam, kde je k jednotlivým elementům dokumentu přistupováno v náhodném pořadí nebo opakovaně.

Standard *DOMu* lze najít v [17].

2.6 Javascript (ECMAScript)

JavaScript je interpretovaný programovací jazyk pro WWW stránky, vkládaný přímo do *HTML* kódu stránky. Jeho syntaxe patří do rodiny jazyků C/C++/Java. Slovo Java je však součástí jeho názvu pouze z marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe. Název jazyku používaný ve standardech je ECMAScript. JavaScript byl původně obchodní název implementace společnosti Netscape. Správný obchodní název pro verzi použitou v prohlížeči firmy Microsoft je JScript.

JavaScript je vkládán do WWW stránek na Internetu, které jsou samy o sobě tvořeny kódem *HTML*. Při prohlížení těchto stránek je JavaScript stažen spolu s WWW stránkou a spuštěn ve webovém prohlížeči. Jsou jím obvykle ovládány různé interaktivní prvky GUI[17] (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků.

Na rozdíl od ostatních jiných interpretovaných programovacích jazyků (např. *PHP* a *ASP*[18]), které se spouštějí na straně serveru ještě před stažením z Internetu, se program v JavaScriptu spouští až po stažení z Internetu (tzv. na straně klienta). Z toho plynou jistá bezpečností omezení. JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele.

2.7 PHP

Název *PHP* vznikl rekurzivním zkrácením Hypertext Preprocessor, původně Personal Home Page. *PHP* je programovací jazyk určený především pro programování dynamických internetových stránek. *PHP* je jazyk interpretovaný, jeho skripty se začleňují přímo do struktury jazyka *HTML* a jsou prováděny na straně serveru. Syntaxe kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). *PHP* je nezávislý na platformě. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, přístup k několika databázovým serverům, podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, …).

PHP se stalo velmi oblíbeným především díky jednoduchosti použití a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou svobodu v syntaxi.

---

[16] Parsování je proces analyzování vstupních dat při zachování gramatické struktury.
[17] GUI – *Graphical User Interface*. V současném technologickém počítačovém prostředí je to označení pro rozhraní mezi člověkem a počítačem.
[18] *ASP* Programovací jazyk. *ASP* je svými možnostmi velmi podobný jazyku PHP. Jedná se o skriptovací jazyk běžící na webovém serveru (Microsoft Internet Information Services).

V kombinaci s databázovým serverem MySQL a webovým serverem Apache je často využíván k tvorbě webových aplikací.

S verzí *PHP 5* se výrazně zlepšil přístup k objektově orientovanému programování. Přístup je velmi podobný jako v jazyce Java.

Od roku 1994 je *PHP* jedním z nejpoužívanějších způsobů tvorby dynamicky generovaných WWW stránek. Jeho tvůrce (Rasmus Lerdorf) jej vytvořil pro svou osobní potřebu přepsáním z Perlu do jazyka C.

2.8 AJAX (Asynchronous JavaScript and XML)

AJAX (*Asynchronous JavaScript and XML*) je technologie určená pro vývoj interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání.
Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, ale vyžadují použití moderních webových prohlížečů.

AJAX aplikace jsou vyvíjeny s využitím technologií:

- HTML (nebo XHTML) a CSS pro prezentaci informací,
- DOM a JavaScript pro zobrazování a dynamické změny prezentovaných informací,
- XMLHttpRequest pro asynchronní výměnu dat s webovým serverem (obvykle je užíván formát XML, ačkoliv je možné použít libovolný jiný formát včetně HTML, prostého textu, JSON či EBML).

Aplikace využívající *AJAX* používají jako platformu pro svůj běh webové prohlížeče, které podporují výše zmíněné technologie. Mezi podporované prohlížeče patří Mozilla Firefox [12], SeaMonkey, Microsoft Internet Explorer [14], Opera [13], Konqueror a Safari.

2.9 Protokol HTTP

HTTP (Hyper Text Transfer Protocol) je Internetový protokol určený původně pro výměnu hypertextových dokumentů ve formátu *HTML*. Používá se HTTP verze 1.1. Standard protokolu je definován v RFC 2616. Tento protokol je spolu s elektronickou poštou nejvíce používaným a zasloužil se o obrovský rozmach internetu v posledních letech.

V současné době je používán i pro přenos dalších informací. Pomocí rozšíření MIME umí přenášet jakýkoli soubor (podobně jako e–mail), používá se společně s formátem *XML* pro tzv. webové služby (spouštění vzdálených aplikací) a pomocí aplikačních bran zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP.

HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků (URL, Uniform Resource Locator), který specifikuje jednoznačné umístění nějakého zdroje v Internetu.

K protokolu HTTP existuje také jeho bezpečnější varianta HTTPS, která umožňuje přenášená data šifrovat a tím chránit před odposlechem či jiným narušením.

2.9.1 Činnost protokolu

Protokol funguje způsobem dotaz–odpověď. Uživatel (pomocí programu, obvykle internetového prohlížeče) pošle serveru dotaz ve formě čistého textu, obsahujícího označení požadovaného dokumentu, informace o schopnostech prohlížeče apod. Server poté odpoví pomocí několika řádků textu popisujících výsledek dotazu (zda se dokument podařilo najít, jakého typu dokument je atd.), za kterými následují data samotného požadovaného dokumentu.

Pokud uživatel bude mít po chvíli další dotaz na stejný server (např. proto, že uživatel v dokumentu kliknul na hypertextový odkaz), bude se jednat o další, nezávislý dotaz a odpověď. Z hlediska serveru nelze poznat, jestli tento druhý dotaz jakkoliv souvisí s předchozím. Kvůli této vlastnosti se protokolu HTTP říká bezestavový protokol – protokol neumí uchovávat stav komunikace, dotazy spolu nemají souvislost. Tato vlastnost je nepříjemná pro implementaci složitějších procesů přes HTTP (např. internetový obchod potřebuje uchovávat informaci o identitě zákazníka, o obsahu jeho „nákupního košíku" apod.). K tomuto účelu byl protokol HTTP rozšířen o tzv. HTTP cookies[19], které umožňují serveru uchovávat si informace o stavu spojení na počítači uživatele.

2.9.2 POST a GET metody

Data mezi stránkami se dají přenášet pomocí metod POST a GET.

Při použití metody GET se veškerá formulářová data předají jako součást URL za otazníkem. Délka URL je 2083 znaků v IE. Při použití metody POST se předají v těle dotazu. Maximální velikost poslaných dat metodou POST je omezena na serveru a může být omezena i na straně klienta[20].

2.10 Graf (teorie grafů)

Graf je základním objektem teorie grafů. Je to uspořádaná dvojice (V, E), kde V je nějaká neprázdná množina a E množina některých dvojic prvků z V.

Prvky množiny V se nazývají *vrcholy* grafu (angl. vertex).

Prvky množiny E se nazývají *hrany* grafu a mohou to být buď uspořádané nebo neuspořádané dvojice (angl. edge).

Řekneme, že prvek *i* sousedí s prvkem *j*, pokud z *i* vede *hrana* do *j*.

---

[19] Cookies se v protokolu HTTP označuje malé množství dat, která WWW server pošle prohlížeči a ten je uloží na počítači uživatele. Při každé další návštěvě téhož serveru pak prohlížeč tato data posílá zpátky na server.

[20] Například v PHP je maximální velikost v konfiguračním souboru php.ini nastavena direktivou upload_max_filesize. Více informací o nastavení PHP viz [9].

2.11 Petriho sítě

Petriho sítě jsou grafický a matematický nástroj vhodný pro modelování a analýzu systémů diskrétních událostí.

Komunikací s automaty se zabýval německý matematik C.A. Petri a ve své disertační práci (1960–62) definoval formalismus později nazvaný Petriho sítě. Tento formalismus byl rozpracován na začátku sedmdesátých let na MIT (Massachusetts Institute of Technology) a od počátku osmdesátých let se každoročně koná mezinárodní konference zaměřená na teorii a aplikace Petriho sítí.

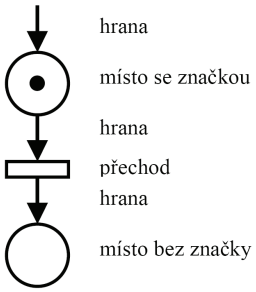
Díky své schopnosti jednoduše modelovat systémy diskrétních událostí obsahující paralelismus, sdílení zdrojů, vazbu na reálný čas a vazbu na okolní prostředí, se Petriho sítě staly oblíbeným nástrojem teoretiků i praktiků. Zejména teoretické výsledky z této oblasti jsou zajímavé, jelikož umožňují formálně prokázat vlastnosti modelovaných systémů.

Petriho sítě lze zdárně aplikovat v různých oblastech informatiky a automatizace. Zvláště jejich schopnost modulárně reprezentovat paralelně běžící procesy a komunikace je výrazně odlišuje od stavového diagramu.

2.11.1 Základní názvosloví

Neoznačená Petriho síť je orientovaný ohodnocený bipartitní graf. Jako každý graf se i Petriho síť skládá z uzlů, jež jsou propojeny hranami. Přívlastek orientovaný značí skutečnost, že hrany grafu jsou orientované. Přívlastek ohodnocený značí skutečnost, že hranám mohou být přiřazeny váhy. Přívlastek bipartitní značí skutečnost, že množina uzlů grafu se skládá ze dvou disjunktních podmnožin – množiny míst a množiny přechodů, přičemž místa a přechody se v průběhu cesty střídají.

Označená Petriho síť vznikne umístěním značek do míst neoznačené Petriho sítě.



Obrázek 1 – Jednoduchá Petriho síť

místo – může obsahovat nezáporný celý počet značek,

přechod – v okamžiku jeho aktivace (přeskoku) jsou odebrány značky ze vstupních míst a přidány značky do výstupních míst přechodu,

orientované hrany – propojují místa a přechody,

počáteční značení (umístění značek v místech před prvním přeskokem) popisuje počáteční stav systému.

Vývoj systému je reprezentován přesunem značek v síti na základě aktivace (přeskoku) přechodů každé nové značení reprezentuje nový stav systému.

Text kapitoly 2.11 je velmi těsně citován z [40].

2.11.2 Petriho sítě s časovanými přechody

Jedním z mnoha rozšíření Petriho sítí (dále jen PS) jsou takové, jejichž funkce je časově závislá. Řadí se do skupiny neautonomních PS a nejčastěji je u nich čas asociován s místy (P–časované) nebo s přechody (T–časované). PS s časovanými přechody lze rozdělit podle typu asociace času na s atomickou aktivací přechodu a s třífázovou aktivací.

Časovaná PS s třífázovou aktivací má k přechodům přiřazeno nezáporné číslo τ udávající dobu trvání přechodu. Pokud je přechod uvolněn, jsou tokeny nutné k aktivaci v jeho vstupních místech rezervovány, tj. nelze je použít k uvolnění nebo aktivaci jiných přechodů. Po uplynutí doby τ od rezervace je přechod aktivován, rezervované tokeny odebrány vstupním místům a naopak přidány místům výstupním, stejně jako u neautonomní PS.

U PS s atomickou aktivací přechodů je každému z nich přiřazeno nezáporné číslo ξ. Pro názornost průběhu odpálení si představme, že má každý přechod svůj lokální časovač. Pokud dojde k uvolnění přechodu, je časovač nastaven na dobu ξ. Hodnota časovače se s přibývajícím časem plynule snižuje. V okamžiku, kdy dosáhne nuly, je přechod aktivován, tj. dojde ke změně značení PS. Pokud je uvolněno současně několik přechodů, jsou nastaveny v jeden okamžik všechny jejich časovače. Jejich dekrementace probíhá pro všechny stejně rychle. Přechod s nejnižším časem se odpálí, čímž deaktivuje všechny přechody, jejichž časovač ještě nedosáhl nulové hodnoty. Tím jsou zastaveny jejich časovače, které se mohou rozběhnout při další aktivaci přechodu, přitom se volí jedna ze dvou strategií:

Časovač se rozběhne z hodnoty, kterou si pamatuje z minulého (přerušeného) odpočítávání.

Časovač je nastaven na hodnotu ξ a nastartována jeho dekrementace.

2.11.3 Stochastické Petriho sítě

Stochastická Petriho síť (SPS) je speciálním případem PS s časovanými přechody T_j, jejichž doby trvání ξ_i jsou náhodné veličiny s exponenciálním rozdělením. Jeho distribuční funkce je

$$F(x) = P(\xi_j \le x) = 1 - e^{-\frac{x}{\mu}}; \quad x > 0 \qquad (1.1)$$

a hustota pravděpodobnosti

$$f(x) = \frac{1}{\mu} e^{-\frac{x}{\mu}}, \qquad (1.2)$$

kde $\mu > 0$ je hodnota $EX = \mu$, převrácená hodnota $\lambda = 1/\mu$ se nazývaná intenzita (poruch).

2.11.4 Stochastické časované Petriho sítě

Stochastická časovaná Petriho síť je Petriho síť s časovanými přechody, které mohou mít dobu trvání nulovou, přesně určenou nebo vyjádřenou pomocí náhodné veličiny. Přitom rozdělení pravděpodobnosti doby trvání u stochastických přechodů je obecné a jeho výběr záleží na konkrétní aplikaci. Uvedu několik nejznámějších rozdělení.

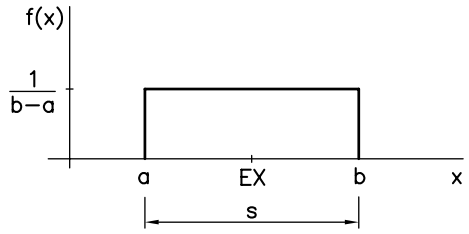
Rovnoměrné rozdělení

Je určeno hustotou pravděpodobnosti

$$f(x) = \frac{1}{b-a} \quad pro \quad a < x < b, \tag{1.3}$$

kde *a* je poziční parametr, *s = (b – a)* měřítko (Obrázek 2), přitom

$a, b \in R, \ a < b$. *EX = (a+b)/2* je střední hodnota.



Obrázek 2 – Hustota pravděpodobnosti *f(x)* rovnoměrného rozdělení

Logaritmicko–normální rozdělení

Náhodná veličina X má logaritmicko–normální rozdělení $LN(\zeta, \sigma^2)$ pokud má hustotu pravděpodobnosti

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{\left[-\frac{(\ln x - \zeta)^2}{2\sigma^2}\right]} \quad pro \quad x > 0, \tag{1.4}$$

kde parametry $-\infty < \zeta < \infty, \ \sigma^2 > 0$ jsou střední hodnota a rozptyl logaritmu náhodné veličiny *Y = ln(X)*, která má normální rozdělení $N(\zeta, \sigma^2)$.

Erlangovo rozdělení

Je odvozeno od Gama rozdělení a určeno hustotou pravděpodobnosti

$$f = \frac{x^{a-1}}{b^a \Gamma(a)} e^{-\frac{x}{b}} \quad pro \quad x > 0, \tag{1.5}$$

kde parametry $b \in R, \ b > 0$ a $a > 0$ je přirozené číslo, $\Gamma()$ je gama funkce (více v [45]).

Rozdělení Rayleigh Tail

Rayleighovo rozdělení se zvětšením (fází) $\sigma > 0$ a spodní mezí $a > 0$ má hustotu pravděpodobnosti

$$f(x) = \frac{x}{\sigma^2} e^{\frac{a^2 - x^2}{2\sigma^2}} \quad pro \quad \mathrm{x} > \mathrm{a}, \quad a, \sigma \in R. \qquad (1.6)$$

2.12 PNML

PNML (*Petri Net Markup Language*) je jazyk popisující Petriho sítě, založený na technologii *XML*. Hlavními rysy PNML je snadná rozšiřitelnost, obecnost, nezávislost na platformě, snadná transformace do jiného formátu a možnost zpracování pomocí standardních nástrojů. Poslední tři vlastnosti jsou důsledkem toho, že je PNML založeno na XML. Obecnost znamená, že lze pomocí PNML zachytit širokou škálu dnes existujících typů Petriho sítí, což plně podporuje k tomu nutná rozšiřitelnost.

Tento formát je standardizován normou ISO: ISO–IEC15909–2.WD.V0.9.0 a lze ho najít na [48].

2.13 StochPNTD

Jedním z požadavků na editor je, aby umožňoval vytvářet grafy odpovídající STPN, k čemuž je nutné disponovat datovým formátem pro její perzistentní uložení. Jelikož žádný takový založený na XML nebo dokonce PNML neexistuje použil jsem StochPNTD[21].

2.13.1 Dodefinované vlastnosti

V definicích základních Petriho sítí byl pouze rozšířen element přechodu *<transition>* o časové vlastnosti. Rozšíření je realizováno pomocí elementu *<time>* (viz. ukázka XML kódu). Typ zpoždění přechodu je dán hodnotou atributu *type*, dle které se současně řídí počet a typ elementů vnořených v *<time>*. Povolené hodnoty obsahuje, pokud není atribut uveden je použita hodnota *zero*. V závislosti na hodnotně jsou povoleny různé podelementy (), např. pokud atribut *type* nabývá hodnoty *stochastic–lognormal* obsahuje element *<time>* elementy *<sigma>* a *<zeta>*, které reprezentují parametry logaritmicko–normálního rozdělení. Samotný údaj parametru je uložen jako atribut *value*.

[21] Odpovídající RNG schéma je v souboru StochPNTD.pntd. Formát StochPNTD lze najít v [46].

```
-<transition id="T1">
    <graphics />
    -<name>
        <text>T1</text>
     </name>
    -<!--  pridane casove vlastnosti prechodu   -->
    -<time type="stochastic-exponential">
        -<mean-time value="0.1">
        -<!-- gracke atributy pro zobrazeni -->
        </mean-time>
    </time>
</transition>
```

Hodnota	Význam – doba trvání přechodu
zero	nulová (použito jako výchozí hodnota)
deterministic	přesně určená (deterministická)
stochastic–uniform	náhodná veličina s rovnoměrným rozdělením
stochastic–lognormal	náhodná veličina s logaritmicko normálním rozdělením
stochastic–exponential	náhodná veličina s exponenciálním rozdělením
stochastic–erlang	náhodná veličina, Erlangovo rozdělení
stochastic–raylegh–tail	náhodná veličina, rozdělení Raylegh tail

Tabulka 1 – Přípustné hodnoty atributu type a jejich význam

Hodnota atributu	Název elementu	Význam (blíže viz. 2.11.4)	Typ hodnoty
zero	žádný		
deterministic	delay	délka zpoždění	NR
stochastic–uniform	mean–time	střední hodnota	NR
	width	šířka pásma	PR
stochastic–lognormal	zeta	střední hodnota logaritmu náhodné veličiny	R
	sigma	rozptyl logaritmu náhodné veličiny	PR
stochastic–exponential	mean–time	střední hodnota	PR
stochastic–erlang	a	tvar	PN
	b	měřítko	PR
stochastic–raylegh–tail	a	spodní mez	PR
	sigma	fáze	PR

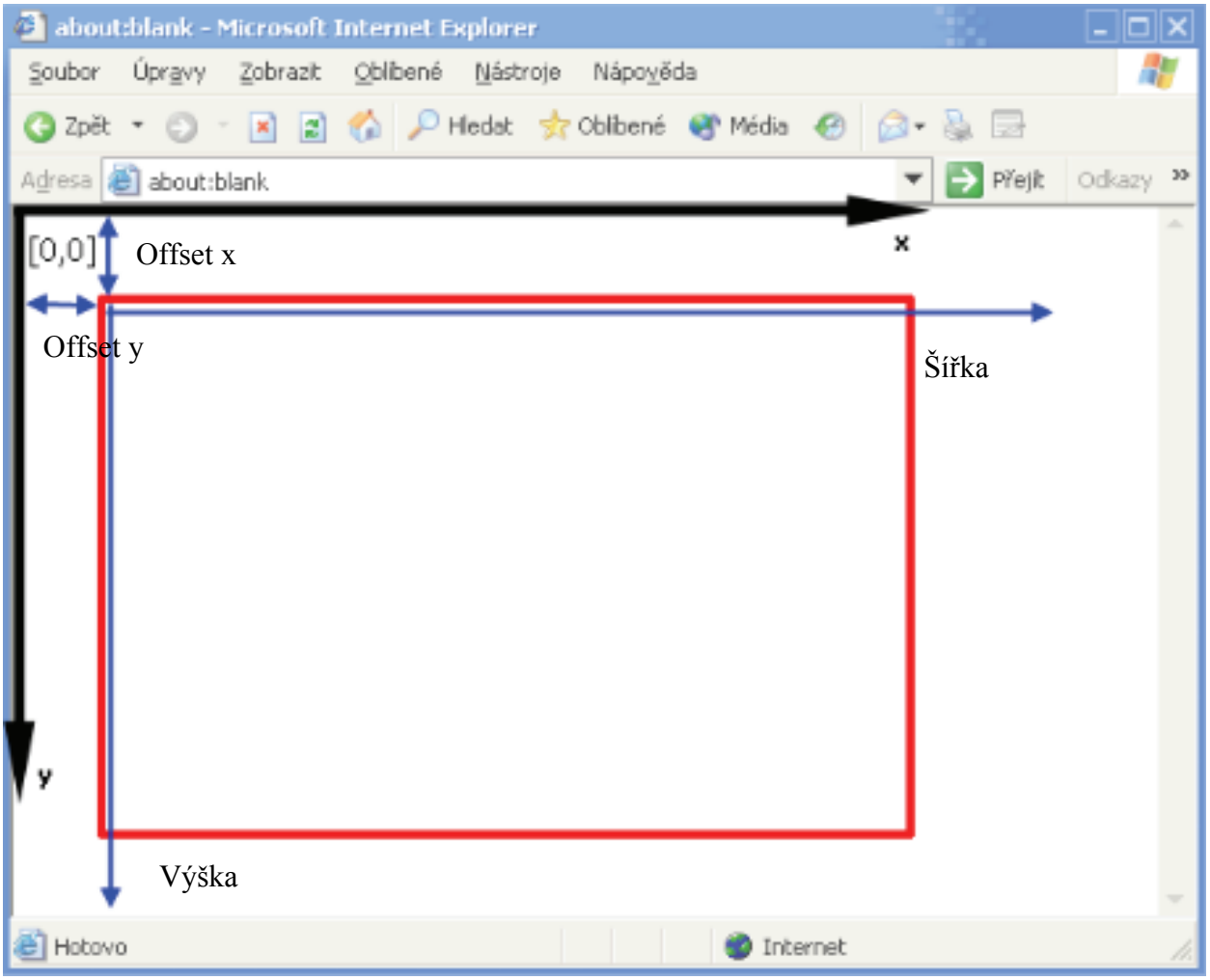
Tabulka 2 – Podelementy elementu <time> v závislosti na hodnotě atributu type.

NR – nezáporné reálné číslo, PR – kladné reálné číslo, R – reálné číslo, PN – celé kladné číslo.

3 Architektura obecného editoru grafů v JS

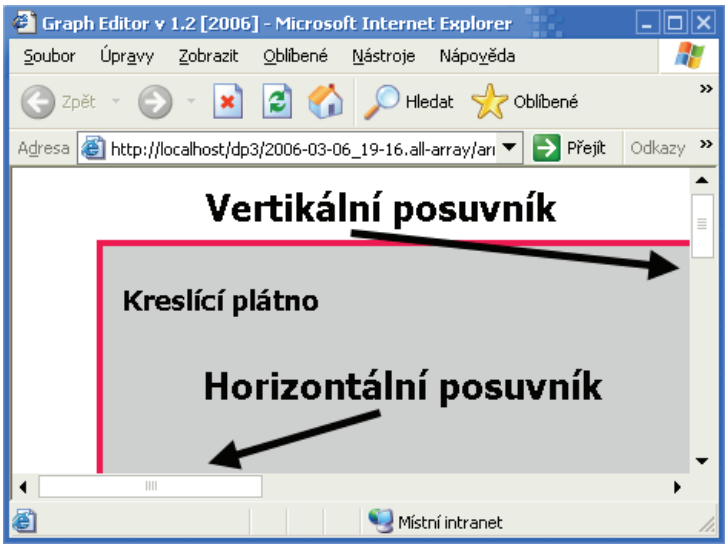
3.1 Kreslící plátno

Plocha v internetovém prohlížeči má kartézské souřadnice [x,y]. Kreslící plátno ve vnitřním okně prohlížeče je posunuto od počátku souřadné soustavy [0,0] umístěném v levém horním rohu okna o [Offset x, Offset y] (viz. Obrázek 3). Šířka a délka kreslícího plátna je libovolně volitelná.



Obrázek 3 – Okno prohlížeče Internet Explorer a kreslící plocha

V případě, že je velikost kreslícího plátna větší než velikost okna prohlížeče, v prohlížeči se automaticky vytvoří horizontální a vertikální posuvník (viz. Obrázek 4).



Obrázek 4 – Horizontální a vertikální posuvníky v prohlížeči

Protože se jedná o editor grafů, kreslené objekty dělíme na tři skupiny.

Na objekt:

- *vrchol*
- *hrana*
- *pomocné objekty* – např. *popisky*

Tyto objekty mají rozdílnou implementaci interaktivity a chování. Např. objekt *vrchol* stačí po vykreslení posouvat v souřadnicích [x,y]. Objekt *hrana* se musí navíc za běhu překreslovat v závislosti na směru a délce. Objekt *popisek* se musí posouvat a lze měnit text popisku.

3.2 Kreslení ve webovém prohlížeči

Funkce pro kreslení ve webovém prohlížeči není standardně implementována. Nakreslit obrázek lze různými způsoby, staticky nebo dynamicky. Statické obrázky jsou vygenerovány při načtení *HTML* dokumentu. Dynamické lze za běhu, pomocí JavaScriptu, zaměňovat za jiné. Kreslit lze také vykreslováním objektů *DIV* nebo pomocí jazyka VML.

Mezi existující způsoby kreslení patří níže vyjmenované metody.

3.2.1 Statické metody kreslení

V tomto odstavci je popsána metoda statického kreslení obrázku.

Do kódu *HTML* je vložen odkaz na obrázek[22]. V případě, že prohlížeč obdrží kód *HTML*, dojde k jeho zpracování a vykreslení obrázku do prohlížeče na místo odkazu. Obrázek je po vykreslení neměnný[23].

[22] Obrázek – nepárový tag. Do stránky se vloží obrázek načtený z jiného souboru. Tag <img> má kromě obecných atributů spoustu dalších. Mezi nejdůležitější patří atribut *src*, který definuje umístění souboru s obrázkem. Umístění je reprezentováno řetězcem URL. Více o URL viz kapitola 2.9.1 .

[23] Obrázek se může měnit např: animovaný GIF, ale nemůže libovolně reagovat na události.

3.2.2 Dynamické metody kreslení

Kreslení pomocí VML

VML je jazyk vyvinutý společností Microsoft, který je určen k tvorbě vektorové grafiky[24]. VML je podporováno v Internet Exploreru 5 a 6, navíc funguje i v sadě Microsoft Office 2000 a vyšší. Avšak VML je již „mrtvým jazykem" a bylo nahrazeno formátem SVG. VML je vázán jen na IE. Prohlížeč IE má sice míru zastoupení okolo 65 až 70 % (květen 2006), ale na zbylých 30–35% prohlížečů by tato metoda nešla použít.

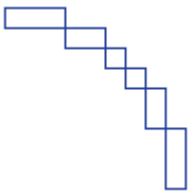
Specifikace jazyka VML byla standardizována u W3C roku 1998. Specifikace lze najít na [1]. Projekt VML editor, který řeší jednoduché kreslení (je funkční jen v IE verze 5.5 a vyšší), se nachází na [2].

Použití knihovny pro generování Divů

Knihovna je naprogramovaná v JavaScriptu. Obsahuje metody pro vykreslení čar, elips, kružnic, čtverců, obdélníků, lomených čar a polygonů, grafických elementů, které nejsou k dispozici v *HTML*. Více informací viz. [3].

Metoda je založena na generování absolutně umístěných objektů *DIV*[25]. Pokud budeme mluvit v dalším textu o objektu *DIV*, máme na mysli objekt *DIV* v objektu *DOM*. Výhoda je, že tuto metodu podporují všechny grafické internetové prohlížeče. Nevýhodou tohoto řešení je, že se pro každý bod vykreslí jeden *DIV* a pokud kreslíme grafické objekty, generují se tisíce až milióny objektů *DIV* poskládaných vedle sebe (příklad krátkého oblouku viz. Obrázek 5), což představuje zátěž, na kterou nejsou prohlížeče stavěny. Tuto metodu nelze použít pro kreslení v reálném čase.

Následující Obrázek 5 ilustruje nakreslení oblouku za použití 6 absolutně umístěných ***DIVů***. Oblouk je 9 pixelů dlouhý a 9 pixelů široký.



Obrázek 5 – Kreslení pomocí objektů DIV

Z projektů, kde je tato knihovna použitá, jmenujme Online Function Grapher – vykreslení průběhu funkcí. Projekt lze vyzkoušet na [4].

[24] Grafika vytvořená pomocí křivek.
[25] DIV je také chápán jako typ oddílu, zahrnuje v sobě libovolně velkou oblast textu včetně nadpisů, obrázků a tabulek.

Kreslení vrcholu pomocí objektu obrázek

Jazyk *HTML* (*DHTML*[26]) poskytuje jen velmi omezené možnosti pro kreslení.

Princip kreslení:

Vrchol je tvořen objektem *DIV* ohraničující obrázek IMG[27]. Obrázek je dynamicky generován pomocí technologie *PHP* (viz. kapitola 3.6), v závislosti na typu a vlastnostech *vrcholu*[28] (viz. kapitola 3.3.1) a může být za běhu změněn na jiný. Změna je uskutečněna výměnou odkazu obrázku pomocí JS. Řešení změny je uvedeno dále v textu v kapitole 3.3.

Objekt *DIV* je absolutně umístěn pomocí stylů[29] na kreslící ploše. Styl obsahuje atributy *left* a *top*. *Left* a *top* jsou souřadnice levého horního rohu kresleného objektu. Hodnoty parametrů stylu objektu *DIV* se mohou za běhu měnit. Nastavení je uskutečněno v JS, pomocí funkce pro změny atributů objektů (jedná se o *W3C DOM Level 1* funkci *setAttribute*). Po změně parametrů se objekt umístí na požadovanou polohu na kreslící plochu.

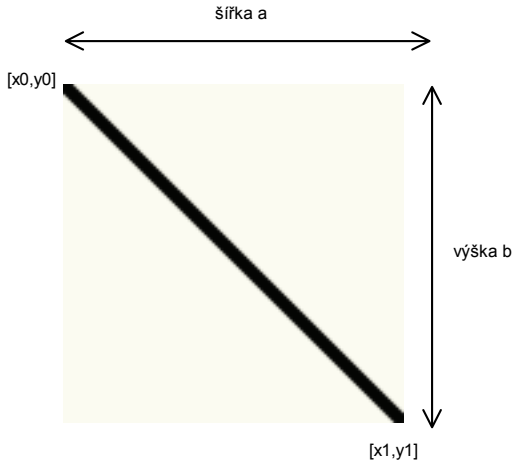
Kreslení hrany pomocí objektu obrázek

Aby byl editor použitelný i pod jinými systémy a prohlížeči, bylo nutné použít řešení změnou atributů obrázku pomocí Javascriptu.

Kód *HTML* obsahuje text. Tvoří ho symboly a pravidla definovaná schématem W3C.

V prohlížeči není naimplementována interaktivita, se kterou by se dala jednoduchými programátorskými technikami nakreslit *hrana*, tak aby uživatel klikl do kreslící plochy a poté tažením myši vybral požadovanou koncovou polohu.

Javascript umožňuje u obrázku změnit atributy – šířku (a), výšku (b) a pozici (x0,y0) – levého horního rohu. Viz. Obrázek 6.

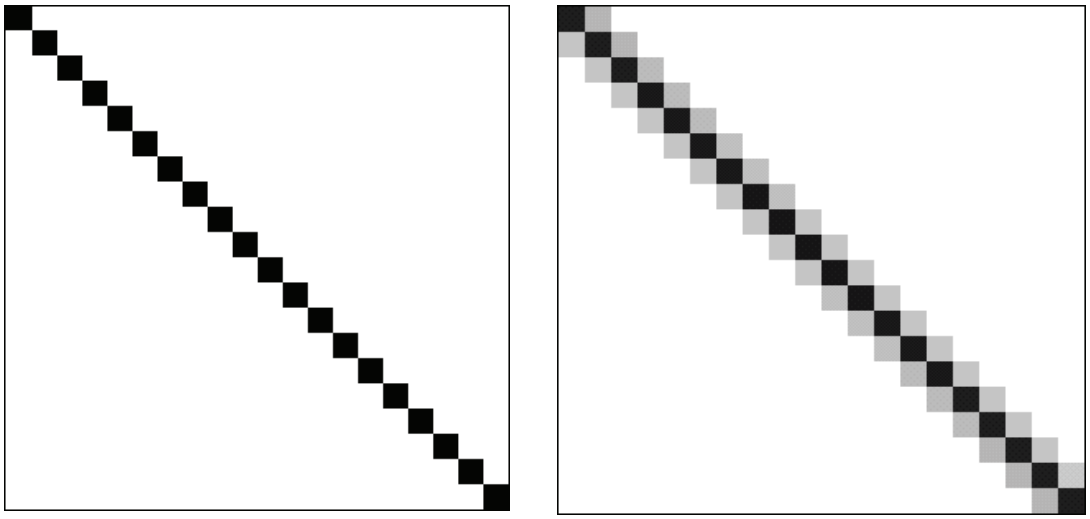


Obrázek 6 – Parametry hrany při vykreslení

[26] Dynamické HTML je kód vzniklý sloučením HTML a CSS.
[27] <IMG> tag je používán pro vložení grafiky do HTML.
[28] Vlastnost vrcholu je např. tvar, barva a další grafické vlastnosti.
[29] Stylistické značky určují vzhled elementu při zobrazení. Více viz. kapitola 2.4.

Vzhledem k tomu, že při změně velikosti obrázku dochází k deformacím a k jednoduchému převzorkování bez antialiasingu, bylo nutné nalézt takový obrázek, u kterého k převzorkování, dojde, ale ztráta kvality bude minimální. Takovým je čtvercový tvar a v něm úsečka vedená od levého horního do pravého spodního rohu.

Obrázek ilustruje rozdíl mezi normálním (vlevo) a antiliasingovaným obrázkem (napravo).



Obrázek 7 – Srovnání aliasing a antialiasingu nakreslené hrany (zvětšeno)

Pokud bych chtěl předejít převzorkování a zachovat kvalitu obrázku, bylo by nejlepší připravit si předgenerované obrázky všech možných kombinací šířky a výšky, které mohou *hrany* dosáhnout.

To bohužel nelze vzhledem k možnému počtu kombinací, které by neúnosně zvýšilo datový tok směrem od serveru ke klientovi a datové náročnosti samotných obrázků. A tak se musím spokojit s jejich daleko menším počtem. Nejlepších výsledků (tj. co nejmenší ztráty kvality a malého datového toku) je možné dosáhnout s velikostí 1 pixelu a násobky dvou. Obrázky obsahují nakreslenou úhlopříčku o tloušťce 1 pixelu.

3.2.3 Metodika měření zkreslení

Odlišnost dvou obrázků (originál a porovnávaný) udává hodnota PSNR (Peak signal–to–noise ratio). Tato metoda se používá pro zjištění kvality rekonstruovaných obrázků.

Měření PSNR

Předpokládejme, že zdrojový obrázek f(i,j) obsahuje NxM pixelů a rekonstruovaný obrázek F(i,j). Chyba je vypočítaná jako rozdíl jednotlivých hodnot pixelů v rozsahu černé (0) a bílé (255). Pixel má celkem 256 odstínů.

Nejdříve vypočteme MSE - střední kvadr. odchylku (mean squared error)

$$\mathrm{MSE}=\frac{\sum_{i=0}^{N-1}\left(\sum_{j=0}^{N-1}\left[f(i,j)-F(i,j)\right]^2\right)}{\mathrm{N}\cdot\mathrm{M}}. \tag{3.1}$$

Spočteme odmocninu z MSE (3.1)

$$\mathrm{RMSE}=\sqrt{MSE} \tag{3.2}$$

a vypočteme hodnotu v dB

$$\mathrm{PSNR}=20\log\left(\frac{255}{\mathrm{RMSE}}\right). \tag{3.3}$$

Pro barevné obrázky je definice shodná až na výpočet MSE, který se určí podle vzorce:

$$\mathrm{MSE}=\sum_{p=0}^{mp-1}\frac{\sum_{i=0}^{N-1}\left(\sum_{j=0}^{N-1}\left[f(i,j)-F(i,j)\right]^2\right)}{\mathrm{mp}\cdot\mathrm{N}\cdot\mathrm{M}}, \tag{3.4}$$

kde *mp* označuje počet barev na pixel (např.: pro obrázky ve formátu RGB – pixel složený ze tří barev RGB, je *mp* je rovno 3).

Typické PSNR je mezi 20 a 40. Hodnoty nejsou významné, ale v porovnání mezi dvěma hodnotami PSNR pro různé obrázky podají míru odlišnosti.

Příklad porovnání:

Porovnával jsem pomocí balíčku ImageMagick [7], určeného pro zpracování a editaci obrázků. Příklad porovnání:

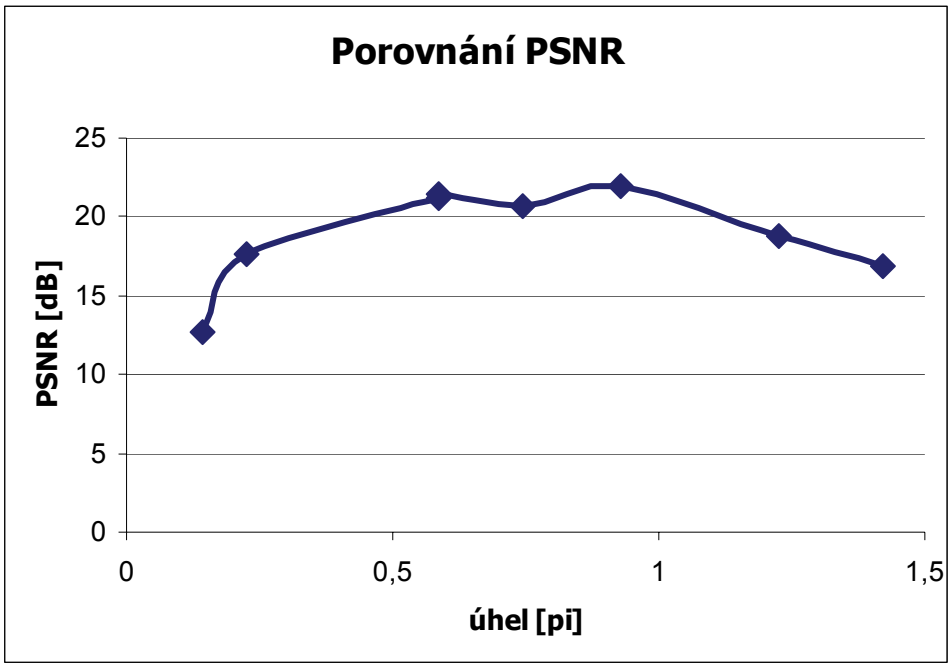
```
compare -metric PSNR soubor1.jpg soubor2.jpg rozdíl.png
  výsledek: 29.5615 dB (a rozdílový soubor rozdíl.png[30])
```

Postup porovnání:

Cílem je porovnat deformované a ideální *hrany* a zjistit, jak moc se liší. Vygeneroval jsem několik *hran*. Každou natočenou o úhel v rozmezí π/8 až π/12 oproti předchozí.

Každá *hrana* měla stejnou délku. Poté jsem vygeneroval ideální *hranu*[31] (tj. *hranu* vedoucí z levého horního rohu do pravého spodního) a porovnal.

[30] Soubor vnikl odečtením prvního obrázku od druhého.



Obrázek 8 – Graf závislosti zkreslení ve srovnání s ideální hranou

Podle grafu na obrázku (Obrázek 8) je vidět, že nejlepších hodnot dosahuje *hrana* u úhlů: $k\pi/2$ ($k \subset N$). U úhlů mezi $k\pi/2$ a $k\pi$ je odchylka největší a dochází k největšímu zkreslení.

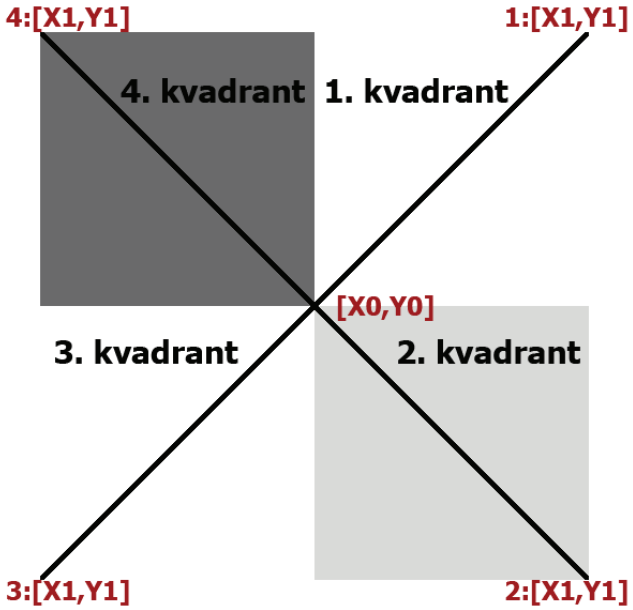
Měření jsem porovnával v intervalu ($0–\pi/2$). Pro intervaly ($\pi/2–\pi$), ($\pi–1{,}5\pi$) a ($1{,}5\pi–2\pi$) je graf shodný.

3.2.4 Nakreslené předgenerované úsečky

K nakreslení mi stačí 2 typy vygenerovaných tvarů. Úhlopříčka s počátkem v levém dolním rohu a končící v pravém horním, nebo začínající v levém horním a končící v pravém dolním rohu, viz obr. Obrázek 21.

Nakreslená úsečka je stejná v 1. a 3. a v 2. a 4. kvadrantu. [X0,Y0] je počáteční bod. [X1,Y1] je koncový bod.

[31] Ideální hranou je myšlena, co nejlepší aproximace na zobrazovací mřížku.



Obrázek 9 – Kvadranty pro kreslení hrany

Z požadavků na kvalitu a rychlost jsem pro editor vybral 18 předgenerovaných obrázků ve formátu GIF[32].

```
1down.gif
1up.gif
2down.gif
2up.gif
4down.gif
4up.gif
8down.gif
8up.gif
16down.gif
16up.gif
32down.gif
32up.gif
64down.gif
64up.gif
128down.gif
128up.gif
256down.gif
256up.gif
```

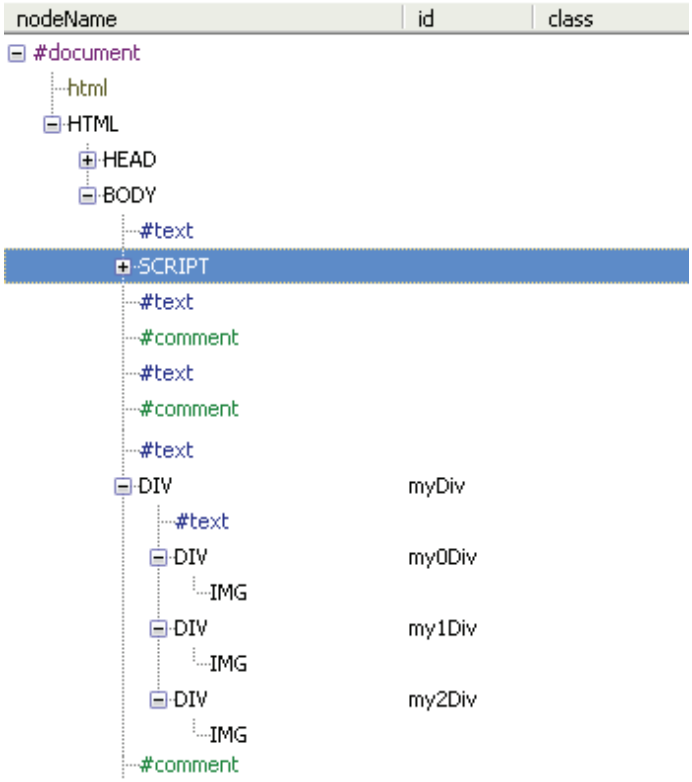
Soubory zabírají celkem cca 3,3 kB. Tuto hodnotu lze bezproblémově přenést ke klientovi. Obrázky se načítají v JS při načtení stránky. Načtením se odstraní nevýhoda posílat vícekrát požadavek během pohybu a čekat v průběhu pohybu na server. To znamená, že při změně velikosti úsečky z 8 na 16 a pak zpátky na 8 by se musely posílat 3 požadavky na server.

[32] GIF (Graphics Interchange Format) je grafický formát určený pro rastrovou grafiku, která se skládá převážně z jednoduchých čárových objektů a ne příliš velkého počtu barev (maximálně 256).

3.3 Objekt vrchol

3.3.1 Algoritmus vytváření objektu

V *HTML* dokumentu hlavní stránky je v sekci body[33] umístěn objekt *DIV* s názvem „MyDiv“.
a) Při vytváření objektu *vrchol*, se projde dokument *HTML* (*DOM* objekt). U *DOMu* je struktura dokumentu reprezentována jako hierarchický objekt viz. obr. Obrázek 10.



Obrázek 10 – DOM struktura dokumentu HTML v prohlížeči

b) Poté se najde objekt *DIV* s názvem (id) „myDiv“ a pokud neexistuje potomek, vytvoří se další vnořený *DIV* do kterého se umístí následující obsah:

```
<img name=\"name\" src=\"newplace.php?
options=yes&type=circle&radius=50&a=50&b=50&tokens=5&thickness=5&cr=20&cg=30&cb=4
0&transp=0&cfr=98&cfg=150&cfb=45\"  border=\"0\">
```

Vnořený *DIV* je dále pojmenován „my0Div“. To je z důvodu vyhledávání konkrétního *vrcholu*. *Vrcholy* jsou pojmenovány my0Div, my1Div, my2Div.. atd.

Vlastnosti *vrcholu* závisí na parametrech (tj. typ *vrcholu*, *hrany* , poloměr, barva…) a jsou přenášeny jako část URL adresy.

Např.:

```
options=yes&type=circle&radius=50&a=50&b=50&tokens=5&thickness=5&cr=20&cg=30&cb=4
0&transp=0&cfr=98&cfg=150&cfb=45
```

[33] Tělo dokumentu obsahuje veškerý zobrazovaný obsah stránky.

type	Označuje tvar *vrcholu* (kružnice nebo obdélník), hodnota je *circle* nebo *rectangle*.
radius	V případě kružnice – poloměr, hodnota je číslo.
A	V případě obdélníku – strana a (horizonální), hodnota je číslo.
B	V případě obdélníku – strana b (vertikální), hodnota je číslo.
thickness	Tloušťka vnější čáry, hodnota je číslo (číslo větší než 1).
cr	Barva obrysu – červená, hodnota je číslo (0–255).
cg	Barva obrysu – zelená, hodnota je číslo (0–255).
cb	Barva obrysu – modrá, hodnota je číslo (0–255).
transp	Transparentní výplň (je–li transparentní, pak je hodnota rovná 1, není–li transparentní hodnota je různá od jedné).
cfr	Barva výplně – červená, hodnota je číslo (0–255).
cfg	Barva výplně – zelená, hodnota je číslo (0–255).
cfb	Barva výplně – modrá, hodnota je číslo (0–255).

Tabulka 3 – Parametry objektu vrchol

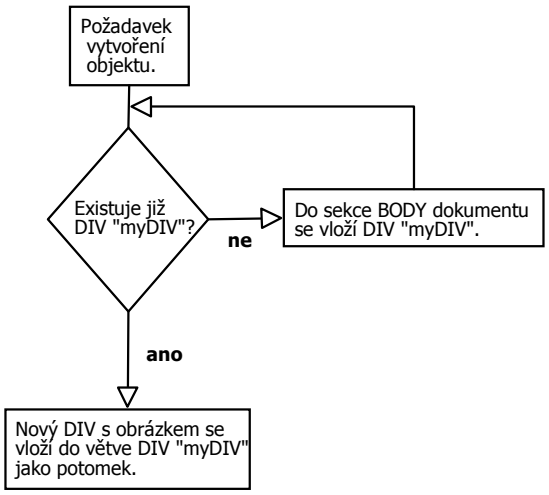
Vlastnosti jsou načítány z definičního *XML* souboru, který definuje grafické vlastnosti objektů. Musí být validní ke Stochastickým definicím verze 0.99. (StochPNTD.rng). Více o definičním souboru viz. kapitola: 3.9.

Nastavení atributů stylu vytvořeného objektu *DIV* se provede následujícím způsobem:

```
//newdiv je nově vytvořený DIV (na začátku „my0Div")
//nastavení id
newdiv.setAttribute("id",divIdName);

//nastavení atributů style, určující pozici
newdiv.setAttribute("style","position:absolute; left:-100px; top:-100px; ");
```

Pozn: záporné hodnoty jsou zde, aby se zaručila kompatibilita s prohlížeči a nedocházelo k „problikávání stránky".

Pokud již nějaký potomek existuje, inkrementuje se proměnná obsahující počet již nakreslených objektů a vytvoří se další potomek. Celý algoritmus je v následujícím vývojovém diagramu.



Obrázek 11 – Vývojový diagram vytváření DIVu

3.3.2 Vnitřní datová struktura objektu vrchol

this.isstart = new Array();	Obsahuje pole proměnných, které udává, zda je počátek *hrany* v objektu nebo ne.
this.bind = new Array();	Obsahuje pole objektů *hran*, s kterými je objekt spojen vazbou. Obsahuje ukazatel na objekt *hrana*.
this.index	Pořadí objektu (kolikátý objekt se vytváří).
this.x	Určuje aktuální polohu v souřadném systému, x je souřadnice levého horního rohu.
this.y	Určuje aktuální polohu v souřadném systému, y je souřadnice levého horního rohu.
this.name	Jednoznačné jméno objektu. Podle tohoto jména lze objekt v *DOMu* vyhledat.
this.centr=new Array();	Obsahuje souřadnice [x,y] s polohou středu objektu.

Tabulka 4 – Vlastnosti objektu vrchol

this.isconn;	True/false. True, pokud je *vrchol* jen pomocný.
this.type	Tvar *vrcholu*: kruh, čtverec, elipsa.
this.radius	Poloměr objektu. (Obsahuje hodnotu, pokud se jedná o kruh.)
this.a	Délka vodorovné strany objektu – používá se při tvaru obdélník.
this.b	Délka svislé strany objektu – používá se při tvaru obdélník.
function addDiv();	Metoda, která vyhledá objekt *DIV* a vloží další objekt *DIV* jako potomka.
function removeDiv();	Odstraní tento *DIV*.
function moveDiv();	Posune objekt *DIV* na zadané souřadnice.
function getX=getX();	Metoda pro zjištění souřadnice x.
function getY=getY();	Metoda pro zjištění souřadnice y.
function getName();	Zjistí jméno objektu.

Tabulka 5 – Vlastnosti objektu vrchol (pokračování)

Objekt *hrana* obsahuje proměnné a metody k vytváření objektu a manipulaci na kreslícím plátně.

Proměnné identifikují parametry objektu. Jedná se o velikost, tvar a souvislost s dalšími objekty – navázanými *hranami*.

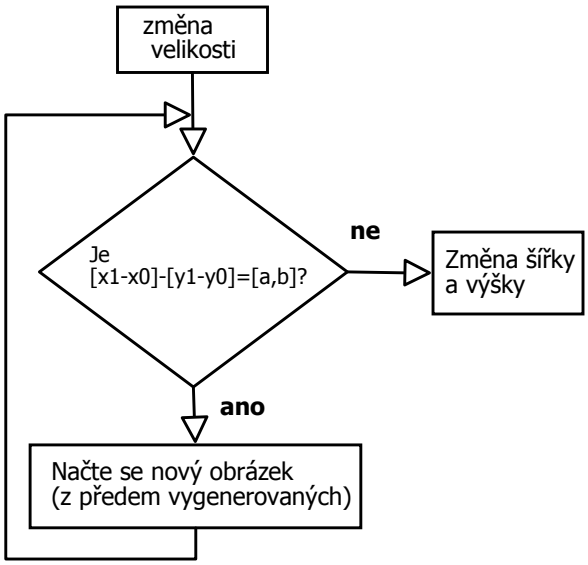
3.3.3 Příklad použití objektu v JS

Pomocí následujícího kódu v JavaScriptu lze vytvořit objekt *vrchol* a vykreslit ho na kreslící plátno. Pro každý objekt je nutno vypočítat střed. Pro objekty tvaru kruh je střed definován jako střed kružnice. Pro obdélníkové jako průsečík os stran. Objekt *hrana* je následujícím kódem nadefinován a lze jej pak posouvat v souřadnicích [x,y] nebo nakonec z kreslící plochy smazat.

```
//nadefinování jména DIVu - přiřazení unikátního identifikátoru
var mdivName="myPic"+actPlaces+"Div";

//vytvoření DIVu s nadefinovaným jménem
aPlaces[actPlaces]=new createDiv(actPlaces,actualx,actualy);

//přidání obsahu do divu
aPlaces[actPlaces].addDiv("<img name=\""+mdivName+"\" src=\"php/
newplace.php?type=circle&radius=100&thickness=5&cr=150&cg=90&cb=150&trans
p=0&cfr=98&cfg=150&cfb=45\" border=\"0\">");

//najde se název objektu
oElement=document.getElementsByName(mdivName);

//objekt se vykreslí na obrazovku a střed objektu bude na pozici kurzoru //myši
aPlaces[actPlaces].moveDiv(actualx-(oElement[0].width/2),actualy-
                (oElement[0].height/2));

//výpočet středu
aPlaces[actPlaces].centr=[actualx-(oElement[0].width/2),actualy-
                (oElement[0].height/2)];
//nastaví se střed
aPlaces[actPlaces].radius=(oElement[0].width/2);
```

3.4 Objekt hrana

3.4.1 Obecný popis



Obrázek 12 – Vývojový diagram změny hrany

Připomeňme, že objekt *hrana* je kreslená jako obrázek viz. kapitola 3.2.2 .

Algoritmus jednotlivých předgenerovaných obrázků:

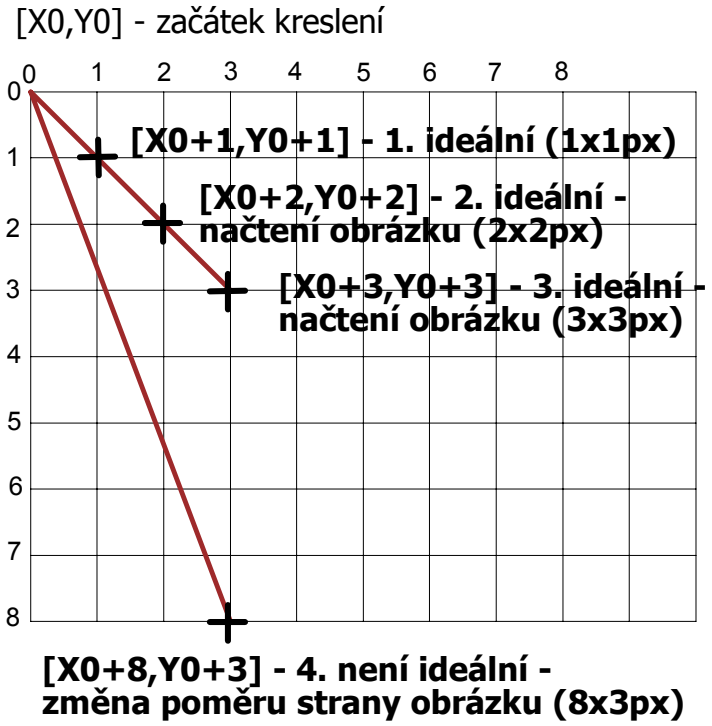
Při změně velikosti se:

1. Nastaví šířka a výška pomocí změny atributů objektu *DIV*.

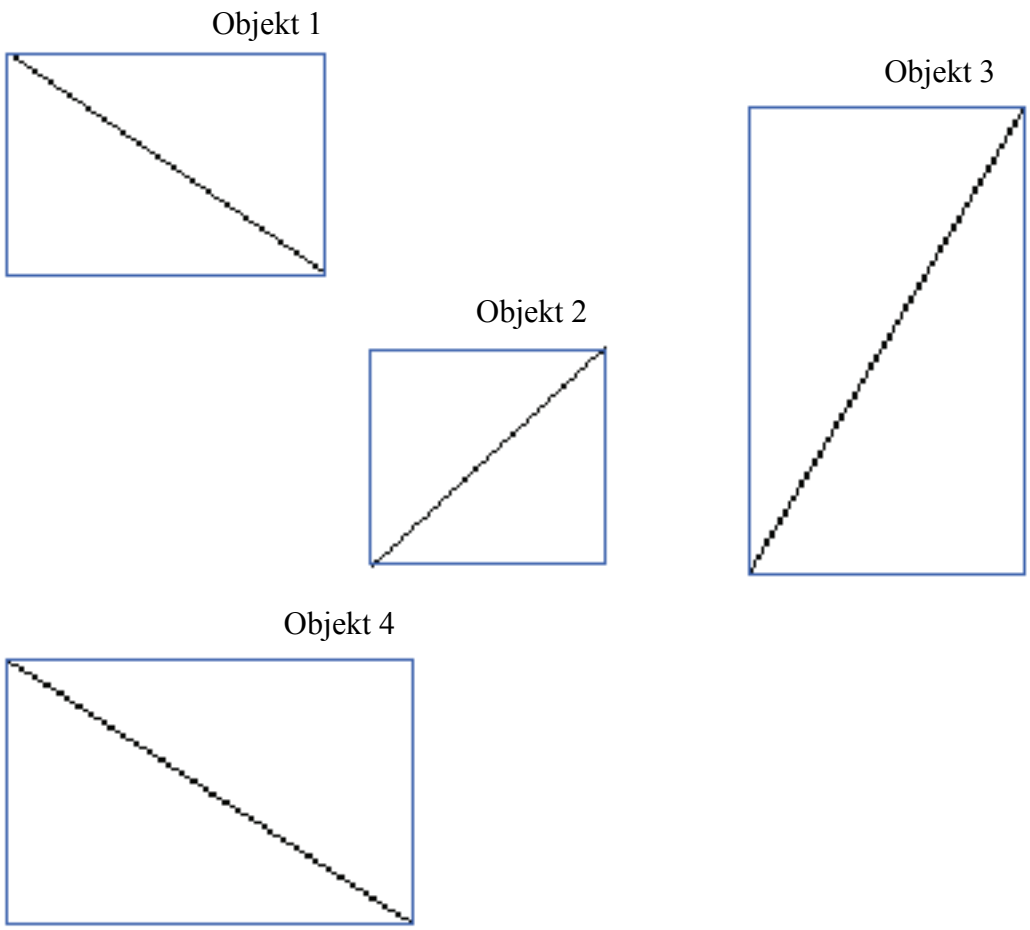
2. Spočítá se rozdíl mezi začátkem a koncem *hrany*. Jakmile je vzdálenost shodná s některým předgenerovaným (viz. kapitola 3.2.4) obrázkem, změní se obrázek na nový.

Použité obrázky se stahují ze serveru najednou při načtení stránky, aby se nemuselo čekat na server při změně, což by způsobovalo nekomfortní ovládání.

Obrázek 13 ilustruje načítání a změnu poměrů obrázku při kreslení hrany.



Obrázek 13 – Schéma kreslení hrany při různých délkách a koncových souřadnicích



Obrázek 14 – Ukázka nakreslených hran

Každá nakreslená hrana tvoří objekt na kreslící ploše. Hrany mají počátek a libovolný konec. Po vložení hrany a určení počátku se délka vybírá následným tažením na kreslící ploše pomocí myši. Obrázek 14 ilustruje nakreslení čtyř hran na kreslící plochu.

Kreslení šipek

Protože v grafu existují orientované *hrany*, bylo nutné nakreslit na konec hrany šipku. Šipka je nakreslená pomocí dalších dvou objektů *hrana*.

Ve vnitřní datové struktuře (viz. kapitola 3.4.2) jsou uloženy ukazatele na dvě hrany tvořící počáteční šipku a na dvě hrany tvořící šipku koncovou.

Obrázek 15 – Nakreslená hrana s koncovou šipkou

3.4.2 Vnitřní datová struktura objektu hrana

Objekt je implementován podobným způsobem jako objekt *vrchol*, až na metody, které obstarávají změnu velikosti objektu a vlastnosti vztahující se jen k tomuto objektu, jako je například šipka.

this.aImgArray;	Pole obrázků, které se mohou načítat při změně velikosti.
this.index;	Pořadí objektu.
this.lineImgPath;	Cesta k načítaným obrázkům.
this.xArray=[x0,y0];	Souřadnice počátku.
this.yArray=[x1,y1];	Souřadnice konce *hrany*.
this.arrowStart;	Obsahuje ukazatel na dvě hrany, které tvoří nakreslenou šipku.
this.arrowEnd;	Obsahuje ukazatel na dvě hrany, které tvoří nakreslenou šipku.
this.lineObjectHandle;	Ukazatel na objekt.
drawLine();	Nakreslí *hranu*.
updateLine();	Změní polohu – aktualizuje.
getID();	Metoda, která zjistí ID – zjistí jednoznačný identifikátor *hrany*.
getXX();	Metoda, která vypíše souřadnice počátku šipky.
getYY();	Metoda, která vypíše souřadnice konce šipky.
deleteLine();	Odstraní *hranu*.

Tabulka 6 – Vlastnosti objektu hrana

3.4.3 Příklad použití objektu v JS

Vytvoření objektu

Vytváření objektu *hrana* je implementováno podobným způsobem jako objekt „*vrchol*“.

Najde se v *HTML* dokumentu v sekci body objekt *DIV* s názvem „myDiv2“ a pokud neexistuje potomek, vytvoří se další vnořený *DIV* do kterého se umístí následující obsah:

Příklad v JS:

```
//proběhne vytvoření objektu
aLines[actLines]=  new Line(actLines,oIn[3].value,
                oIn[4].value,oIn[5].value,oIn[6].value,"img/",aImgArray);
//objekt se vykreslí na obrazovku
aLines[actLines].drawLine();
//proběhne aktualizace
aLines[actLines].UpdateMyLine();
```

Změna objektu při pohybu

Změna pohybu je realizována změnou atributu stylu *DIVu*. Atributy stylu je možno měnit za chodu tímto způsobem:

Příklad v JS:

```
//do objektu zaneseme nové souřadnice
aLines[actLines].setXY();
//proběhne aktualizace
aLines[actLines].UpdateMyLine();
```

3.5 Popisek

Popisek slouží pro identifikaci *hrany* nebo *vrcholu*. Generování popisku probíhá dynamicky.

Vložení textu probíhá obdobným způsobem jako vložení *vrcholu*. Vygeneruje se objekt *DIV* a jako vlastnost se mu nastaví „text“. Do objektu se vloží vygenerovaný obrázek .Obrázek obsahuje atribut „*sr*“ s následujícím textem.

```
drawtext.php?text=NejakyText
```

Vlastnosti popisku

V grafu nesmí docházet k nedefinovatelnosti popisků. To znamená, že každý *popisek* musí mít nějakého vlastníka. Vlastník může být *vrchol* nebo *hrana*.

Text *popisku* se může za běhu editovat. K tomu slouží *PHP* skript pro generování popisků, viz. kapitola 3.6.2 .

3.6 PHP soubory pro generování grafických objektů

Pokud chcete mít na svých stránkách grafický ukazatel často se měnících dat, je řešením generování obrázků pomocí *PHP*. Grafická knihovna GD, kterou používá *PHP*, umožňuje generovat obrázky ve formátech PNG[34], JPEG, WBMP a GIF[35].

Pro generování jsem použil PNG formát. Výhoda formátu je, že nabízí více barev a lepší kompresi než GIF. Navíc obsahuje osmibitovou průhlednost (tzv. alfa kanál), to znamená, že obrázek může být různě průhledný. Komprese JPEG není vhodná pro perokresbu, zobrazení textu nebo ikonky, protože kompresní metoda JPEG vytváří v takovém obrázku viditelné a rušivé artefakty.

3.6.1 Generování vrcholů

Způsob kreslení, který jsem použil, je založeno na generování obrázků. Obrázky generuje *PHP* skript na straně serveru.

Obrázky se generují s transparentním pozadím, aby v případě nakreslení kruhu nevzniklo barevné (nebo bílé) ohraničení, které by mohlo při více objektech pod sebou zakrývat některé vlastnosti.

Obrázek 16 – Vlevo šedivé pozadí vrchního kruhového vrcholu,objekt vpravo má transparentní pozadí

V případě, že na server přijde požadavek na server na *PHP* skript (požadavek obsahuje parametry obrázku), vygeneruje se obrázek podle parametrů.

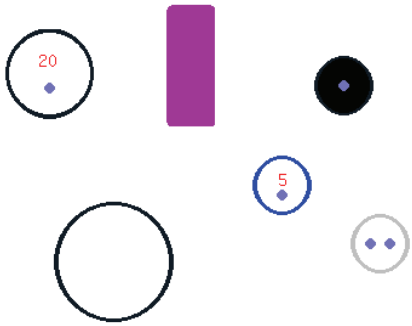
Například následující požadavek vygeneruje kružnicí o poloměru 50 pixelů.

```
newplace.php?options=yes&type=circle&radius=50&a=50&b=50&tokens=5&thickness=5&cr=
20&cg=30&cb=40&transp=0&cfr=98&cfg=150&cfb=45
```

Vysvětlení jednotlivých parametrů je uvedeno v kapitole 3.3.1 .

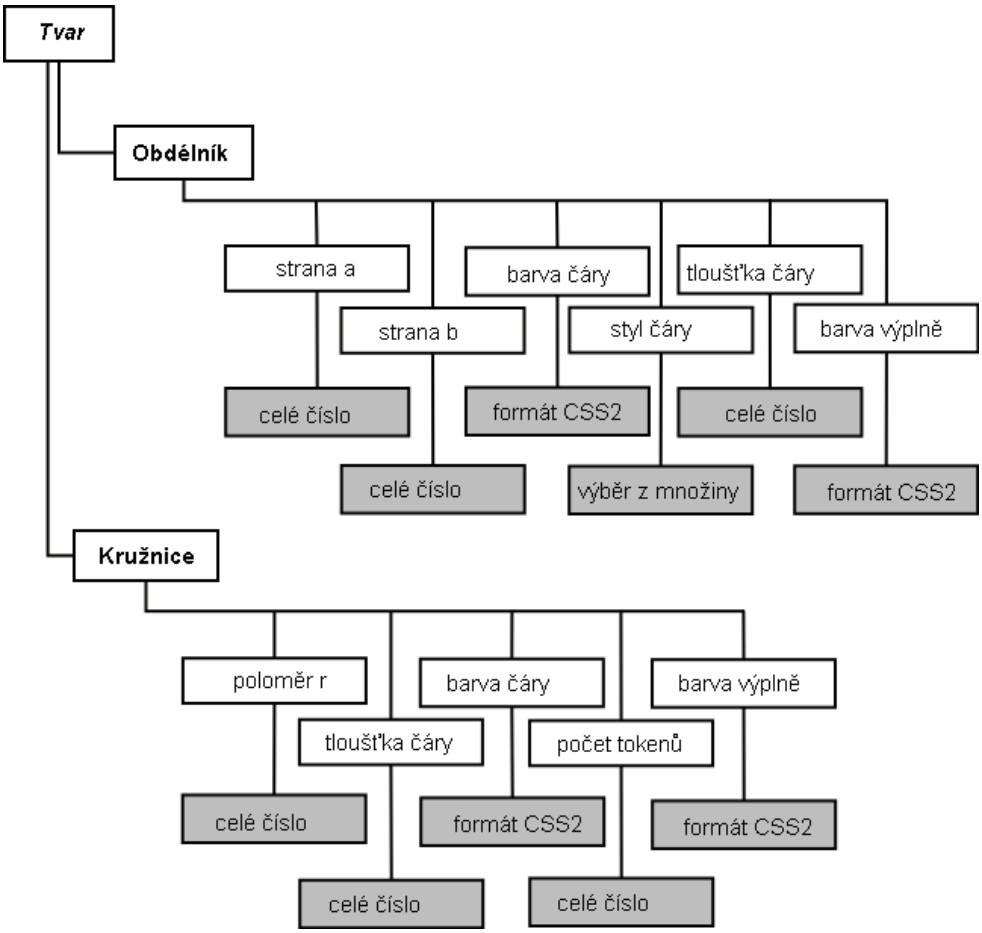
[34] PNG (Portable Network Graphics je grafický formát určený pro bezeztrátovou kompresi rastrové grafiky. Byl vyvinut jako zdokonalení a náhrada formátu GIF, který je patentově chráněný.
[35] GIF (Graphics Interchange Format) je grafický formát určený pro rastrovou grafiku, ale počet použitých barev nemůže přesáhnout 256.

Tento *PHP* skript `newplace.php` umožňuje nakreslit: obdélníkový tvar nebo kružnici. Viz. obr. Obrázek 17.



Obrázek 17 – Příklady nakreslených objektů s různými grafickými parametry

Struktura *PHP* souboru a generování je na obrázku (Obrázek 18).



Obrázek 18 – Struktura PHP souboru pro generování vrcholů

3.6.2 Generování textu

Pro nakreslení grafu je vhodné *vrcholy* a *hrany* popsat. Tento skript se zavolá u objektu *popisek*. Generování textu probíhá podobným způsobem jako generování vrcholů.

Schéma generování:

1. Při změně nebo vytvoření nového popisku se pošle dotaz text na *PHP* soubor na server ve tvaru URL:

```
text.php?text=text?font=1
```

2. *PHP* soubor vygeneruje obrázek.

Generování textu na straně serveru probíhá:

a) *PHP* soubor přijme text (pomocí metody GET).

b) Zjistí se délka textu.

c) Vygeneruje se obrázek s délkou a šířkou odpovídajícími textu.

3. Dojde k načtení nového obrázku (tento krok je vlastností *HTML*).

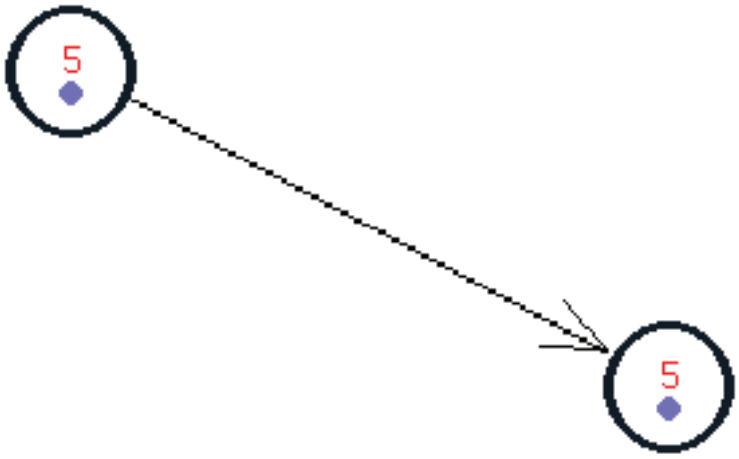
3.6.3 Generování hran

Připomeňme, že z důvodu významných technických omezení použitých technologií při kreslení je *hrana* generována jako převzorkovaný obrázek (viz. kapitola), u kterého dochází k aliasingu.

K odstranění tohoto jevu jsem vygeneroval *hranu* jako obrázek v *PHP* souboru, pomocí *imageSmoothLine* viz. [27].

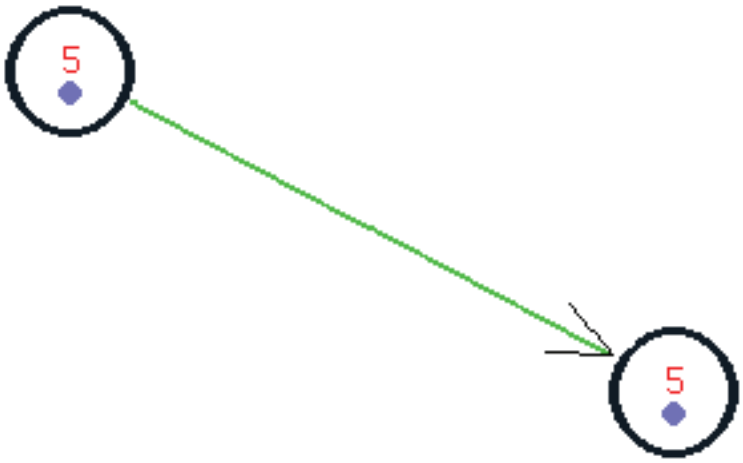
Po změně *hrany* (po dokreslení) se zavolá upravená *hrana* (*imageSmoothLine*) s parametry [x0,y0] – počáteční a [x1,y1] – koncový bod přímky a vygeneruje se nový obrázek. Ten se pak pomocí JS vymění s původním.

Obrázek ilustruje stav před výměnou:



Obrázek 19 – Hrana před výměnou obrázku JavaScriptem

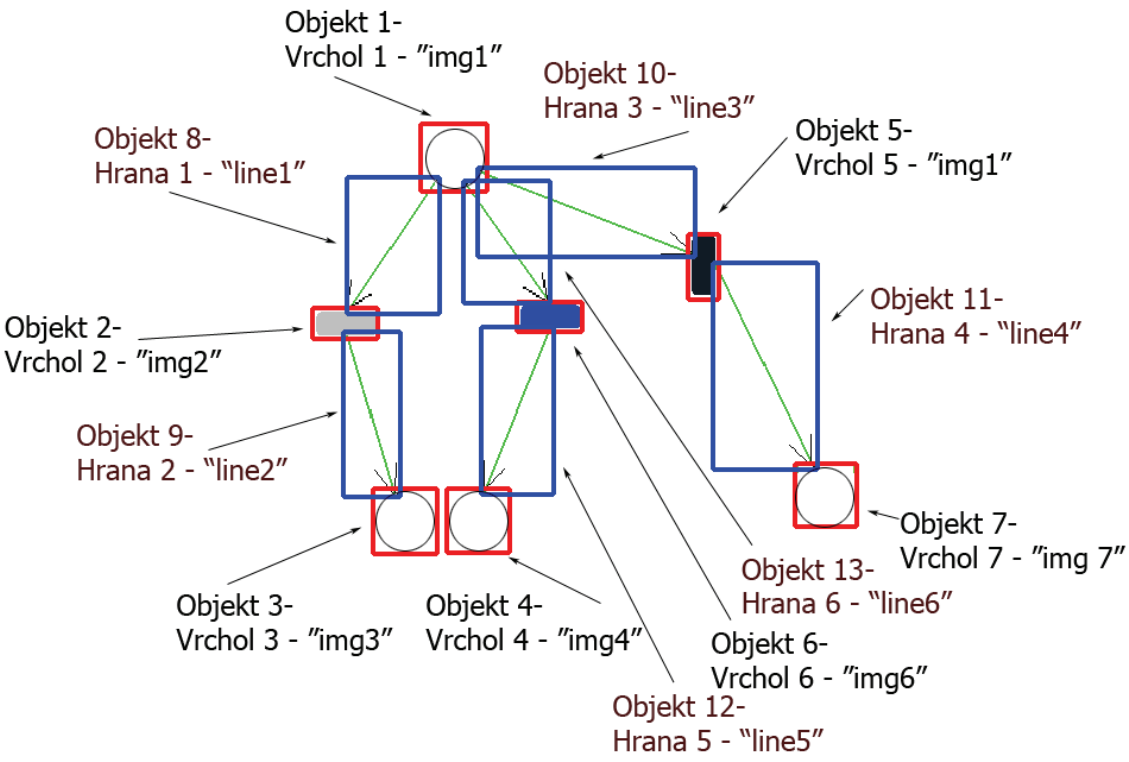
Obrázek ilustruje stav po výměně:



Obrázek 20 – Hrana po výměně obrázku JavasSriptem

3.7 Grafické jádro editoru

Grafické jádro se stará o propojení objektů a jejich správné vykreslení. Objekty na kreslící ploše jsou rozmístěny podle obrázku (Obrázek 21).



Obrázek 21 – Rozvržení objektů na kreslící ploše

Každý objekt je tvořen jedním *DIVem* a obsahuje:

- *vrchol*
- *hranu*
- *popisek*

Objekty jsou vykreslovány na kreslící plátno v absolutních souřadnicích.

3.7.1 Datová reprezentace grafu:

Vkládání objektů vrchol:

V ECMAskriptu (JavaScriptu) je objekt definován jako neuspořádaná množina vlastností, která může obsahovat hodnotu, objekt nebo funkci. Přesněji řečeno, znamená to, že objekt je pole neuspořádaných hodnot.

Z tohoto důvodu jsem se rozhodl nevytvářet objekt *Graf,* ale pole objektů. V grafické jádře editoru zavedu proměnnou, která mi určuje počet již nakreslených objektů.

V případě, že přijde požadavek na vytvoření dalšího objektu, proměnná se inkrementuje a objekt se uloží do asociativního pole.

```
Pole[počet_objektů]=NovýObjekt
```

Instance objektů jsou uloženy v poli:

Objekt[1]	Objekt[2]	Objekt[3]	Objekt[4]	…

Obrázek 22 – Struktura objektů typu vrchol

Ke každému objektu pak mohu přistupovat libovolně.

Vkládání objektů hrana:

Podobná situace je i s objektem typu *hrana*:

Objekt[1]	Objekt[2]	Objekt[3]	Objekt[4]	…

Obrázek 23 – Struktura objektů typu hrana

S tím rozdílem, že při vytváření objektu *hrana* je implementováno několik omezení a pravidel:

1. Objekt musí někde začínat, tj. musí být kliknuto do objektu *vrchol*.
2. Do objektu *vrchol* se uloží objekt *hrany* – začátek.
3. Tažením myši se vybere požadovaná velikost.
4. *Hrana* končí v okamžiku, kdy je kliknuto do objektu *vrchol*.
5. Do objektu *vrchol* se uloží objekt *hrany* – konec.

Tímto způsobem je vytvořena datová struktura, která umožňuje pohyb objektů se závislostmi.

Pohyb objektů:

V této podkapitole bude vysvětlen pohyb objektů.

Pokud je vybrán objekt *hrana*, lze objektem pohybovat ve směrech x a y.

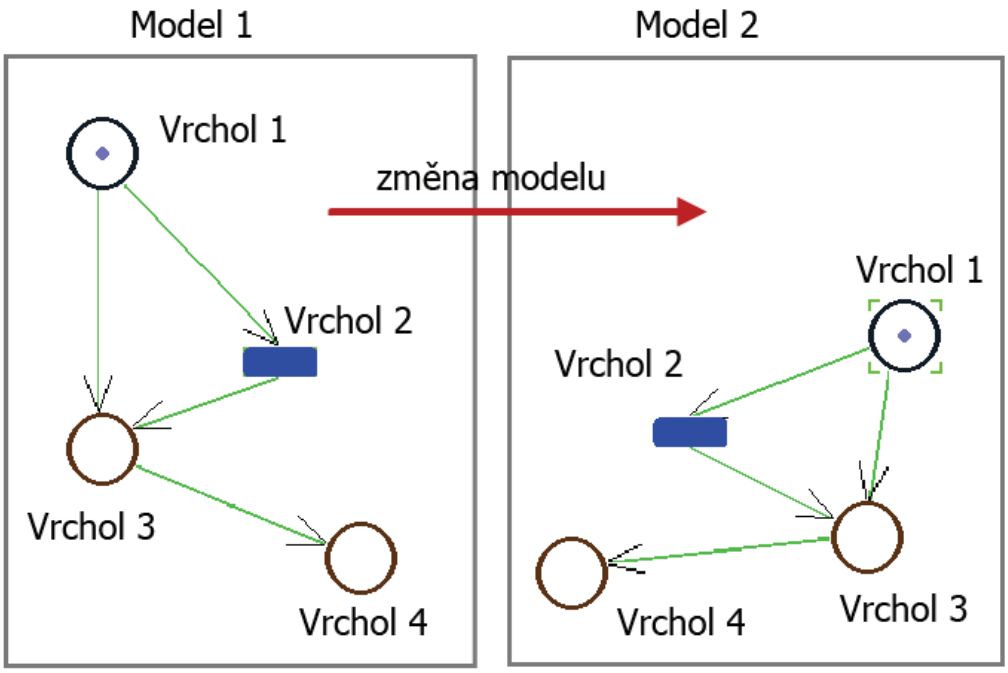
Postup:

Kliknutím se vybere objekt a zjistí se typ zavoláním `objekt.tvar`.

Pokud je tvar kružnice, použije se funkce pro výpočet průsečíku kružnice–úsečka, pokud je vybrán obdélníkový tvar, vybere se funkce pro počítání průsečíku obdélník–úsečka (viz. kapitoly 3.7.2 a 3.7.3).

U vybraného objektu zavolám metody pro změnu polohy. V případě, že k *vrcholu* náleží *hrana* (nebo více *hran*), jsou nalezeny, zjistí se typ propojených objektů (kruh, čtverec, obdélník) a počet *hran*, vypočítá se průsečík s okrajem zvoleného tvaru.

Pohyb ilustruje následující obrázek:



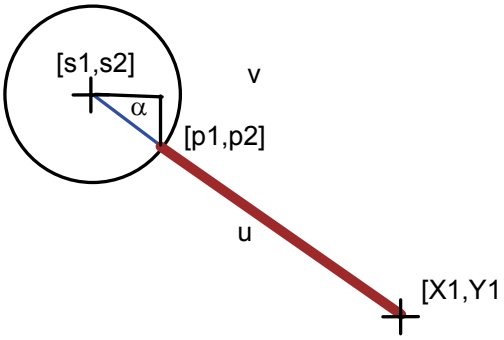
Obrázek 24 – Obrázek ilustruje výsledný graf (Model 2) po pohybu vrcholů v souřadnicích [x,y]

3.7.2 Funkce pro výpočet průsečíku kružnice–úsečka

Tato funkce počítá průsečík kružnice s úsečkou.

Funkce má parametry: poloměr, střed kružnice, koncový bod úsečky, která má počátek ve středu kružnice.

Výsledkem je vypočtený průsečík o souřadnicích [p1,p2].



Obrázek 25 – Výpočet průsečíku kružnice s úsečkou

Průsečík se počítá pomocí goniometrických funkcí.

3.7.3 Funkce pro výpočet průsečíku obdélník–úsečka

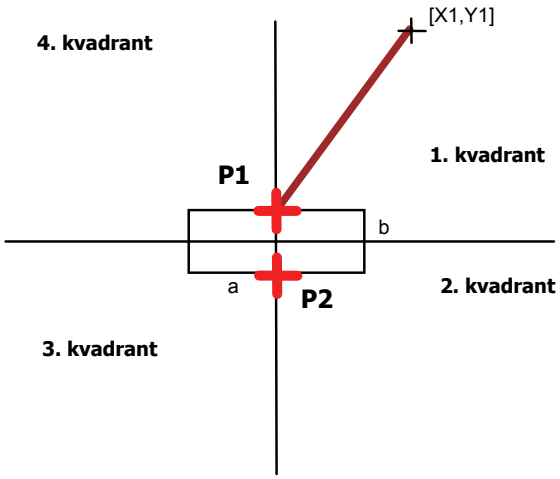
Tato funkce počítá průsečík obdélníka s úsečkou.

Funkce má parametry: stranu a, stranu b, střed obdélníku, koncový bod úsečky, která má počátek ve středu kružnice.

Výsledkem je vypočtený průsečík P1 nebo P2 (příp. P3 a P4) v závislosti na poměru délky a šířky.

Při výpočtu mohou nastat tři možnosti:

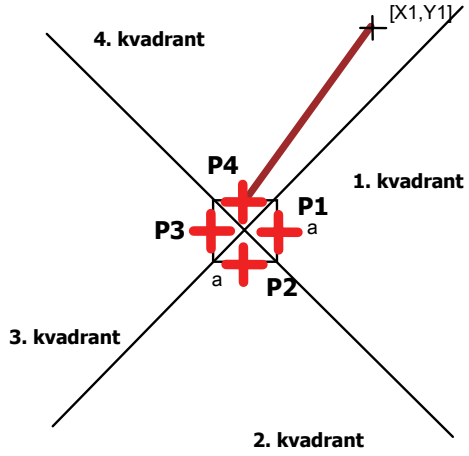
1) **Strana a>b**



Obrázek 26 – Počítání průsečíku obdélníku strana (a>b) a úsečky

V případě, že je delší strana obdélníka vodorovně, vypočítá se průsečík v kvadrantech 1 nebo 4 na bod P1. V případě, že bod [X1,Y1] – koncový bod úsečky leží ve 2. nebo 3. kvadrantu, výsledkem je bod P2. Bod P2 leží v polovině úsečky strany a. Bod P1 je osově souměrný podle osy strany b.

2) Strana a=b

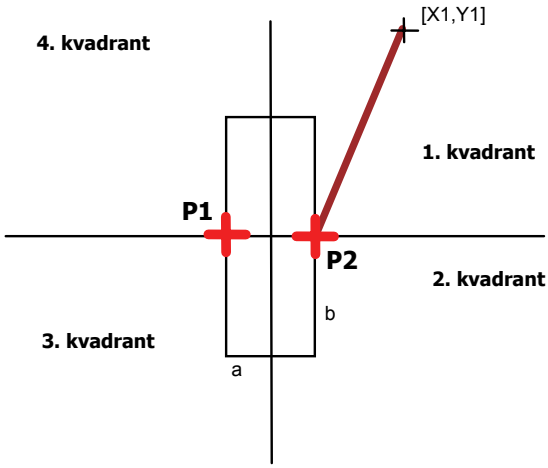


Obrázek 27 – Počítání průsečíku úsečky se čtvercem

V tomto případě má objekt tvar čtverce o straně a. Prostor jsem uhlopříčně rozdělil na 4 kvadranty.

V případě, že se koncový bod úsečky nachází v prvním kvadrantu, vypočte se průsečík jako bod P1, v případě, že ve 2. kvadrantu jako bod P2, v případě, že ve třetím vypočte se bod P3 a v posledním 4. kvadrantu bod P4. Body P1, P2, P3, P4 leží na průsečíku os stran *a* a hrany čtverce.

3) Strana a<b



Obrázek 28 – Počítání obdélníku strana (a<b) a úsečky

Obdélník postavený na kratší základnu. Výpočet probíhá obdobným způsobem jako v předchozích případech. Zde u 1. nebo 2. kvadrantu se vypočte jako průsečík bod P2 a ve 3. nebo 4. kvadrantu bod P1. Bod P2 leží v polovině úsečky strany b. Bod P1 je osově souměrný podle osy strany a.

3.7.4 Mazání objektů

Objekty *vrcholy*, *hrany* a *popisky* jsou uloženy v poli. Vytvořil jsem pro ně stejné algoritmy mazání.

Postup: Po kliknutí se vybere objekt a zjistí se jméno objektu.

V asociativním poli se vybere objekt, který chci odebrat.

Zavoláním metody odeberu objekt.

```
Pole_Hran[vybraná_hrana].deleteLine();
```

A vymažu záznam z asociativního pole.

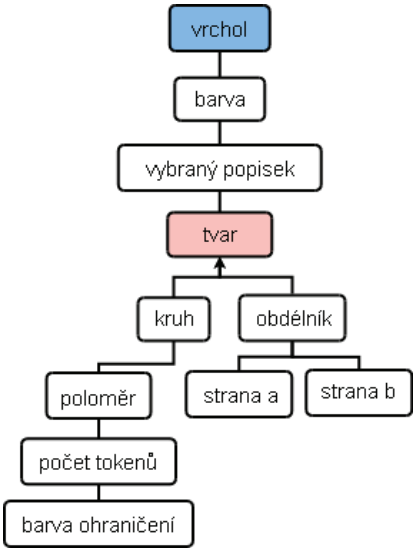
3.7.5 Změna vlastností vrcholu

Každý objekt *vrchol* má definovanou událost *on click* (po kliknutí do objektu) na kreslící ploše. Po kliknutí do objektu se vyberou vhodné vlastnosti. Pro každý typ objektu je definována množina vlastností, které se mohou se měnit.

Vlastnosti objektu jsou uloženy v URL jak parametry GET dotazu, jak již bylo uvedeno v kapitole 3.3.1 .

Změna parametrů probíhá výměnou jednoho nebo více parametru v URL řetězci a následným posláním URL *PHP* skriptu generující grafické objekty.

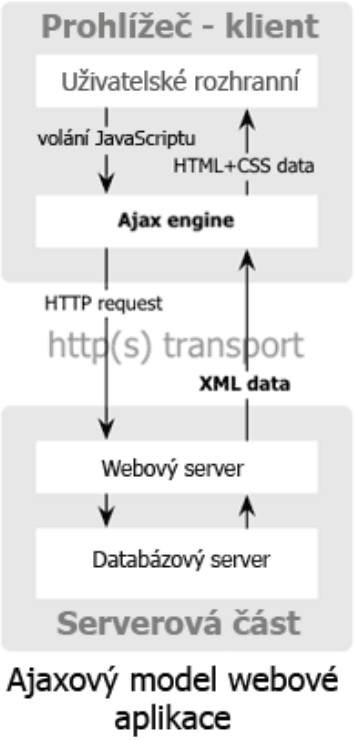
Následující obrázek ilustruje vlastnosti jednotlivých objektů:



Obrázek 29 – Vlastnosti kreslených objektů

AJAX – Asynchronní komunikace se serverem

Technologie *AJAX* slouží pro komunikaci směrem od klienta (prohlížeč) na server (webový server).



Obrázek 30 – Obecné schéma aplikace využívající AJAX technologii, převzato a přeloženo z [42]

Tradiční řešení spočívá v tom, že akce je realizována běžným odkazem nebo formulářem a po jejím provedení se opět nově nahraje celá stránka. Nevýhody jsou zřejmé – celá stránka se musí načíst znovu..

AJAX je technologie určená pro vývoj interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání[36].

Webových aplikací založených na *AJAXu* je celá řada, jedná se například o:

- **AJAX kalendář** – plánovací kalendář pro ukládání, editování, přesouvání a mazání událostí (úkolů, schůzek a poznámek) [29]
- **Basecamp** – program pro spolupráci, plánování a řízení projektů [30]
- **Gmail** – rozhranní webové pošty [31]
- **The Web Word Procesor** – Webový textový editor [32]
- **gOffice** – Webové kancelářské programy – aplikace textový editor, tabulkový procesor a nástroj pro vytváření prezentací [33]
- **Meebo** – IM webová aplikace pro připojení do sítě ICQ, Yahoo, MSN, Gtalk nebo Jabber [34]
- **Flickr** – Online aplikace pro správu a sdílení fotografií [35]

[36] Při znovunačítání stránky se zruší stávající obsah a do prohlížeče se vygeneruje nový.

- **Seznam našeptávač** – pomáhá navrhovat hledaná slova nejen při fulltextovém vyhledávání na internetu [36]
- **Hra Tetris** – V prohlížeči běžící hra Tetris [37]

3.7.6 Použití AJAX technologie

Při použití technologie dynamického generování obsahu stránky v prohlížeči existuje možnost ztráty celého nakresleného grafu.

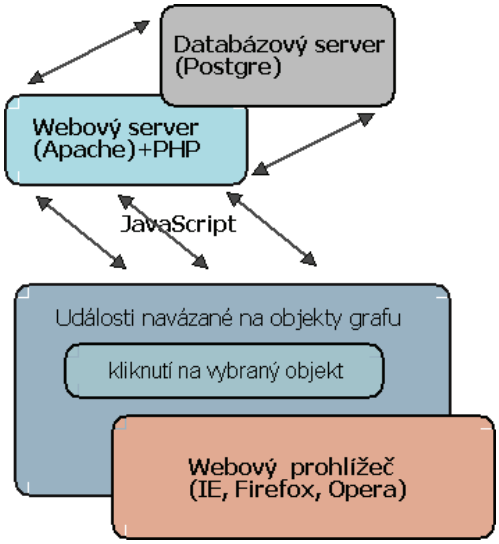
Technologie AJAX znemožňuje použití tlačítka „zpět“ v prohlížeči (tlačítko se používá jen pro statické stránky). Toto se dá bez váhání označit za největší problém AJAXu a dynamicky vytvářených stránek. Uživatel se vrátí na předcházející stránku a aplikace se nevrátí do předcházejícího stavu. Stisknutím tlačítka „zpět“ ztratí uživatel práci, kterou na stránce pomocí AJAXu a dynamického generování objektů vytvořil.

V prohlížeči neexistuje jednotný a spolehlivý způsob jak zabezpečit nestisknutí tlačítka „*zpět*“ a „*obnovit*“ pomocí JS nebo *HTML* do původního stavu. Proto jsem propojil technologií *AJAX* aplikaci s databází. Tímto řešením dosáhnu uložení grafu před pádem prohlížeče.

Hlavní komponentou pro vývoj *AJAX* aplikací je objekt XMLHttpRequest. Při poslání požadavku se nejdříve na straně klienta vytvoří objekt XMLHttpRequest, do objektu se předají parametry metody GET nebo POST a URL požadavku. Po zavolání metody objektu *send* se vyšle požadavek na webový server. Výsledkem je, že je požadavek následně zpracován na http serveru (Apache) a výsledná stránka je vrácena ke klientovi.

Jak jsem zmínil v kapitole 3.2.2 kreslení probíhá dynamickým vytvářením objektů *DIV*. Pozice a vlastnosti jednotlivých *DIV* objektů jsou uloženy v prohlížeči. Představme si situaci, při které dojde k pádu prohlížeče nebo k stisknutí tlačítek „zpět“ nebo „obnovit“. Nastane stav, který známe při náhlém uzavření aplikace – dojde ke zrušení celého nakresleného grafu. Nakreslený graf lze však vrátit do původního stavu, protože se za běhu pozice a vlastnosti jednotlivých objektů průběžně ukládají do databáze.

3.7.7 Technologie řešení



Obrázek 31 – Technologie zpráv mezi klientem a serverem

Front–end[37] editoru spolupracuje se serverem a jedná se tedy o klient – server aplikaci. Na serveru se nachází webový server (Apache) a SQL databáze (PostgreSQL).

Funkce pro posílání dat (parametrů) technologií *AJAX* jsou vázány na události (kliknutí myši na vybraném objektu a tažení objektu po stisknuté levého tlačítka). Jedná se o tyto události:

- Nakreslení nového *vrcholu,*
- nakreslení nové *hrany,*
- pohyb *vrcholu* (a je–li na *vrchol* navázána nějaká *hrana*, pak i pohyb *hrany*),
- změna vlastností *vrcholu,*
- vložení *popisku*,
- pohyb *popisku*,
- změna textu *popisku*.

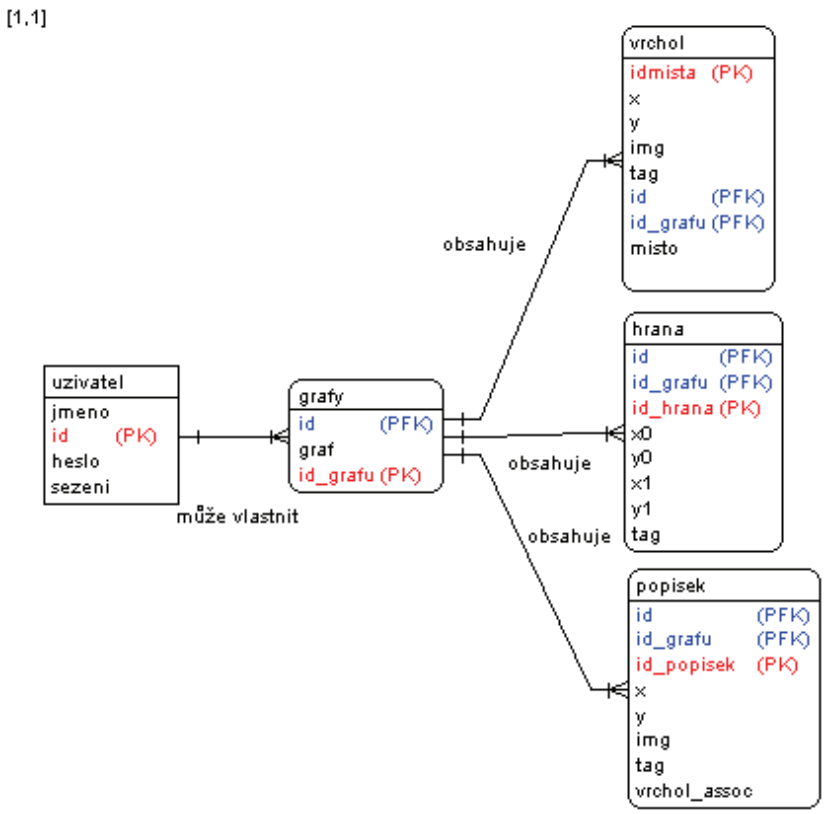
3.7.8 Schéma SQL databáze pro ukládání dat za běhu

Data jsou uložena na serveru v relační[38] databázi.

Po změně (parametrů a umístění) objektů se odesílají pouze data o aktuální pozici objektů a parametrů – synchronizuje se stav v prohlížeči se stavem na serveru v databázi.

Pro ukládání dat jsem použil relační databázi, jejíž struktura je viz. Obrázek 32.

[37] Front–end je viditelné uživatelské rozhraní ve webovém prohlížeči.
[38] Relační databáze je databázový systém, který je založen na relačním modelu dat a relační algebře. Data jsou uspořádána do tabulek (relací), nad kterými jsou definovány přípustné operace.



Obrázek 32 – Schéma databáze

Databáze se skládá z pěti tabulek:

Tabulka *Uživatel* obsahuje identifikátor (*id*) uživatele. Sloupec *jmeno* obsahuje přihlašovací jméno nutné pro přihlášení do systému. Sloupec *heslo* obsahuje MD5[39] hesla a *sezení* je přiřazen vygenerovaný náhodný klíč. Pro každého uživatele, který se chce přihlásit a pracovat s editorem grafů je generováno jméno, heslo a sezení. Tím je uživatel definován a může pokračovat dále.

Tabulka *Grafy* obsahuje počet grafů. Platí tvrzení, že každý uživatel může kreslit více grafů a vybírat si aktuální graf. Sloupec *Graf* obsahuje jméno grafu.

Následující tabulky mají za úkol ukládat data z kreslení a s využitím těchto dat generovat výstupní XML soubor grafu.

V Tabulce *Vrchol* jsou ukládány jednotlivé kreslené *vrcholy* a jejich parametry. Konkrétně: pozice x a y vzhledem k souřadnému systému a atributy obrázku.

Tabulka *Hrana* obsahuje identifikátor uživatele, počáteční a koncovou pozici hrany a její atributy.

Tabulka *Popisek* obsahuje text popisku, jeho umístění a parametry.

[39] Hašovací funkce je předpis pro výpočet kontrolního součtu (haše) ze zprávy či většího množství dat. Může sloužit ke kontrole integrity dat, k rychlému porovnání dvojice zpráv, indexování, vyhledávání apod.

3.7.9 Načítání grafu

Pokud dojde k zavření nebo zrušení okna prohlížeče, nastane v prohlížeči ztráta nakresleného grafu. Tuto vlastnost jsem odstranil průběžným ukládáním grafu a načtením uloženého grafu.

Načtení grafu je možné dvěma způsoby.

Načtením rozpracovaného grafu

Data jsou ukládána za běhu. Po zrušení je lze graf obnovit. Pro vytváření každého grafického objektu jsem vytvořil funkci, která objekt vykreslí.

Je-li vybrána varianta načíst rozpracovaný graf, pak se pomocí *PHP* se při načtení stránky v prohlížeči vygeneruje Javascriptová funkce, která umožní generovat objekty se zadanými parametry. Generování probíhá spuštěním *PHP* skriptu, který se připojí do databáze, zjistí parametry objektů a napíše funkce pro vykreslení *vrchol*, *hran* nebo *popisků* na kreslící plochu.

Generování probíhá odesláním SQL dotazu např.:

```
SELECT * FROM  vrchol where (id=$id and id_grafu=$id_grafu) ORDER BY idmista ASC
```

Kde proměnná *$id* a *$id_grafu* specifikují uživatele a pořadové číslo editovaného grafu.

Získaný výsledek dotazu se vloží do vygenerované JavaScriptové funkce *createPlaceMass()*, kterou se spustí generování míst.

a) Nahrání grafu z XML souboru od klienta na server

Generování JavaScriptových funkcí pro vytváření grafických objektů a připojení Javascriptového souboru za běhu je složitější.

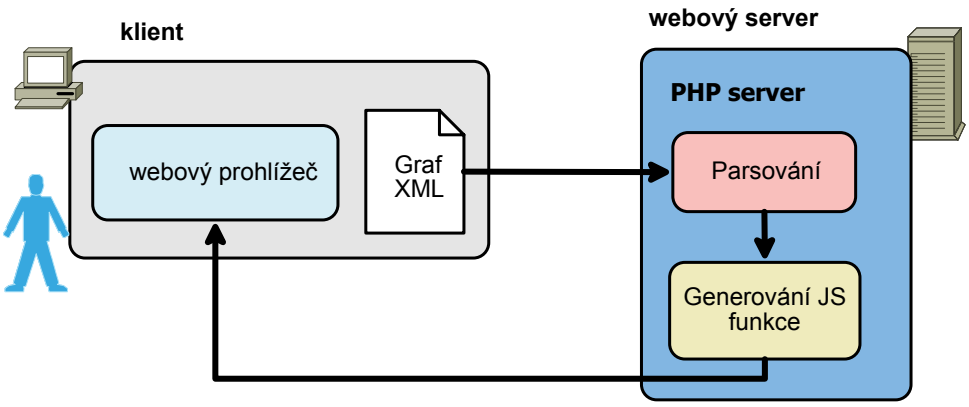
Z tohoto důvodu jsem použil *SAJAX framework* [38], které implementuje vzdálené volání a připojení funkcí JavaScriptu bez obnovení stránky prohlížeče.

Parsování *XML* souboru v JavaScriptu není standardní záležitostí. Parsování *XML* implementují prohlížeče IE (pomocí ActiveX objektu a proprietární syntaxe) a Mozilla (obsahuje *XML DOM* objekt), ostatní tuto funkci neovládají.

Algoritmus nahrání grafu do aplikace (Obrázek 33):

- Uživatel nahraje soubor na server,
- na straně serveru v *PHP* se soubor pomocí funkcí *PHP* SimpleXML[40] naparsuje a dynamicky se vytvoří JavaScriptový soubor,
- Javascriptový soubor se dynamicky připojí do aplikace,
- soubor je připraven k vykonání z aplikace.

[40] SimpleXML slouží pro jednodušší manipulaci s XML jako PHP objektem.



Obrázek 33 – Vygenerování a připojení JS souboru pomocí technologie SAJAX

3.8 Generování XML souboru

Jak jsem již zmínil v úvodu, *DOM* je objektově orientovaná reprezentace *XML* dokumentu (viz. kapitola o *DOM* 2.5). Lze pomocí něj vytvářet uzly, data, atributy a další prvky *XML*. Implementace *DOM* v *PHP* je standardní s rozhranním definovaném v *W3C DOM Level 2*.

Příklad vytvoření jednoduchého *XML* souboru pomocí *DOM* API v *PHP*:

```
<?php
$doc = new DOMDocument('1.0','utf-8');

$root = $doc->createElement('pnml');
$root = $doc->appendChild($root);

$head = $doc->createElement('net');
$head = $root->appendChild($head);

$name= $doc->createElement('name');
$name = $head->appendChild($name);

$value= $doc->createElement('value');
$value = $name->appendChild($value);

$text = 'Jméno grafu';
$p_text = $doc->createTextNode ($text);
$p_text = $value->appendChild ($p_text);

echo $doc->saveXML();
?>
```

vygeneruje následující *XML* soubor

```
<?xml version="1.0" encoding="utf-8" ?>
<pnml>
  -<net>
    -<name>
      <value>Jméno grafu</value>
    </name>
  </net>
</pnml>
```

Všechny informace o grafu jsou uloženy v databázi, v tabulkách *vrcholy*, *hrany* a *popisky.* Skladba PNML souboru je strukturovaná a proto je vhodné výstupní *XML* soubor generovat z databáze.

Na požádání klienta se vygeneruje *PHP* skriptem seznam (pomocí SQL dotazu, obdobným jako v kapitole 3.7.9) a pro ten se pak vygenerují jednotlivé větve ve stromu *DOM*.

3.8.1 Struktura StochPNTD

StochPNTD je šablona k napsání definičního souboru obecného grafu. Bohužel z důvodu omezení technologií jsou implementovány vlastnosti uvedené v kapitolách 3.2 a 3.6.

Kořenovým elementem dokumentu je element *<pnml>*, který obsahuje jednu nebo více Petriho sítí jako element *<net>*. Každá síť má definovaný jednoznačný identifikátor (atribut *id*) a typ (atribut *type*), který je URI jejího schématu. Uvnitř elementu sítě je její název *<name>*, seznam vrcholů *<place>*, přechodů *<transition>* a hran *<arc>*. Název sítě je popisek, může být zobrazen a měněn uživatelem. Obsahuje svůj text jako hodnotu elementu *<text>* a grafický popis v elementu *<graphics>*, který se skládá z pozice textu *<offset>*, fontu, jeho barvy a pozadí nápisu.

V následujícím příkladu je nakreslen jednoduchá Petriho síť obsahující dvě místa a jeden přechod.

```
-<pnml>
  -<net id="n1" type="http://www2.info.../ptNetb.pntd">
  -<net>
     -<name>
       <value>Jmeno grafu</value>
      </name>
  -<place id="p0">
     -<graphics>
       <position  x="225" y="299" />
     </graphics>
  </place>
  -<place id="p1">
     -<graphics>
       <position  x="545" y="296" />
     </graphics>
  </place>
  -<transition id="t1">
     -<graphics>
        <position  x="398" y="299" />
     </graphics>
  </transition>
  <arc  id="arc0" source="p0" target="t1"/>
  <arc  id="arc1" source="t1" target="p1" />
  </net>
  </net>
 </pnml>
```

Místo

Místo je reprezentované elementem *<place>*. Hodnota jeho atributu *id* je identifikátor místa, který je použit pro propojení s hranou. Vnořené elementy jsou *<graphics>* pro popis pozice vrcholu a barvy, *<initialMarking>* obsahující popisek, jež představuje počáteční značení a *<name>* jakožto název vrcholu.

Přechod

Přechod je zapsán jako element *<transition>*, který je až na chybějící značení shodný s elementem místa.

Hrana

Hrana je tvořena elementem *<arc>*, který je nositelem informace o tom jaký přechod a místo propojuje.V elementu místa nebo přechodu to zaznamenáno není. Samotný údaj nese atribut *source*, který obsahuje identifikátor výstupního místa nebo přechodu, a atribut *target* obsahující ID protějšku opačného typu. V podelementu *<graphics>* je uložen seznam pozic mezilehlých kontrolních bodů hrany (*<position>*). První je vždy bod následující za tím, který

je umístěn ve výstupnímu vrcholu. Váha hrany je zapsána jako hodnota elementu *<text>* vnořeného do elementu *<inscription>*.

3.9 Podpůrné definice

3.9.1 Úvod

Strukturu, způsob grafické reprezentace a formát uložení grafu určují tzv. podpůrné definice, které jsou založeny na Relax NG doplněného o elementy z pomocného prostoru jmen.

Editor, z důvodu velkých technických omezení nepodporuje všechny vlastnosti.

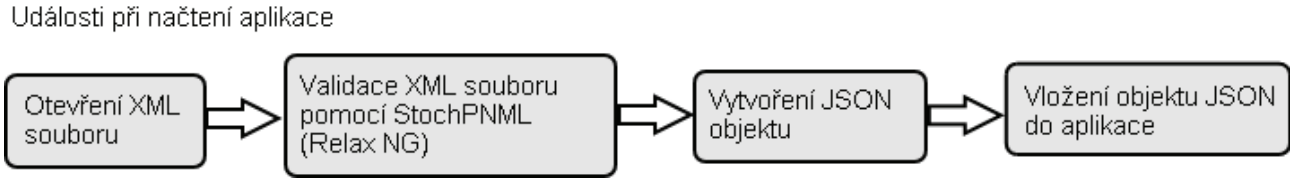
3.9.2 Technologie parsování definičního souboru

Parsování *XML* formátu, jak jsem již zmínil v kapitole 3.7.9 je v internetovém prohlížeči na straně klienta velice obtížné. Také velikost *XML* souboru je v textové formě objemná, a aby byl editor co nejméně náročný na datový tok od serveru směrem ke klientovi, není reprezentace dat v *XML* formátu vhodná.

JSON (JavaScript Object Notation) [39] je odlehčený formát čitelný v textové formě.
V mnohém předčí datový formát *XML*. Pro použití v JavaScriptu je použití datového formátu JSON výhodnější než *XML*, protože parsování datové struktury je rychlejší a je podporováno nativně.

Je založen na JavaScriptu a standardu ECMA a použít jej lze v mnoha programovacích jazycích.

Popis algoritmu:



Obrázek 34 – Popis algoritmu parsování definičního souboru

3.9.3 Propojení s klientem

Po parsování XML souboru podpůrnými definicemi dojde ke vložení JSON objektu do JavaScriptového souboru. Generování probíhá při načtení stránky.

```
<?xml version="1.0" encoding="utf-8" ?>
-<sd:graphics>
    -<sd:skin id="ver1">
      <sd:base-point x="0" y="0" />
        <sd:ellipse a="30" b="20" x="0" y="0" />
    </sd:skin>
</sd:graphics>
```

A vygeneruje se následující řetězec reprezentace objektu typu JSON.

```
"sd:graphics" : {0 : {"sd:skin" : {0 : {"attributes" : {"id" : "ver1"}, "sd:base-
point" : {0 : {"attributes" : {"x" : "0", "y" : "0"}}}}}}}
```

A dalším zpracováním objektu *JSON* v JavaScriptu se vygeneruje řetězec s atributy vrcholu.

Například:

```
type=circle&radius=50&a=50&b=50&tokens=5&thickness=5&cr=20&cg=30&cb=40&transp=0&c
fr=98&cfg=150&cfb=45
```

Řetězec (nebo řetězce) se pak vloží jako parametr při vytváření *vrcholů*.

Konkrétně se výše uvedeným řetězcem vygeneruje kružnice o poloměru 50 pixelů, 5 tokeny, tloušťka čáry 5 pixelů a barvou kružnice ve formátu RGB (barva v rozsahu [0–255] červené 20, zelené 30, modré 40), výplň je transparentní.

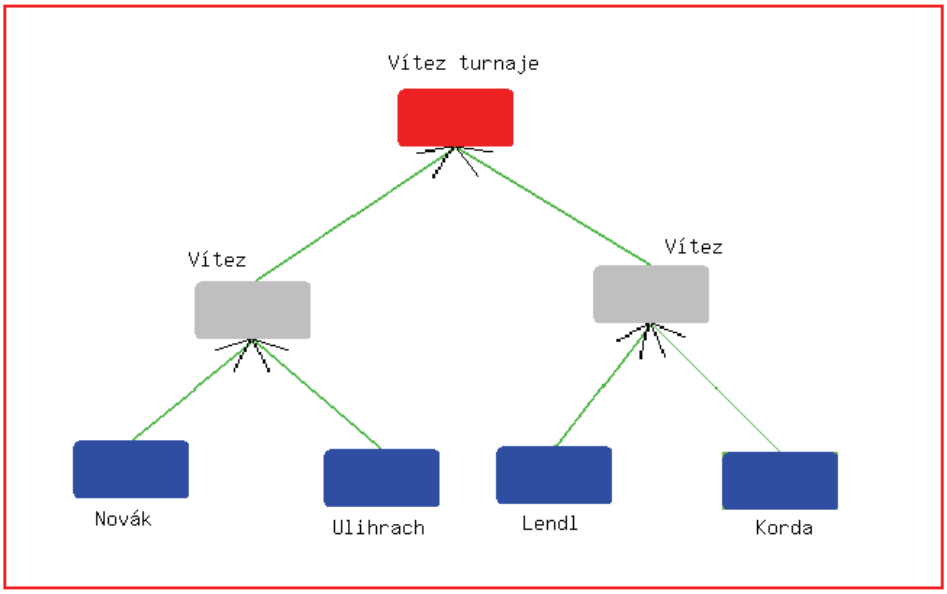
Podrobné vysvětlení jednotlivých parametrů je uvedeno v kapitole 3.3.1 .

4 Specializace editoru

Editor dovoluje použít definiční XML pro specializaci a definic jednotlivých vrcholů.

4.1 Organizační diagram

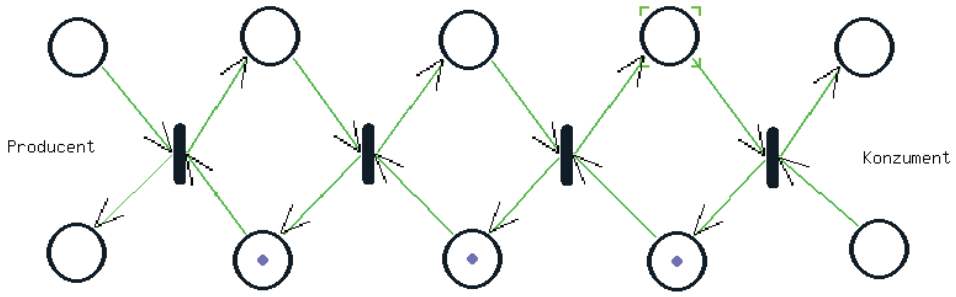
V editoru lze nakreslit organizační diagram turnaje. Diagram je znázorněn na obrázku (Obrázek 35).



Obrázek 35 – Organizační diagram turnaje

4.2 Petriho sítě

Za použití definic pro PS, lze nakreslit Petriho sít. Síť je uvedena na obrázku (Obrázek 36).



Obrázek 36 – Schéma Petriho sítě

5 Závěr

Cílem mé práce bylo vytvořit editor pro stochastické sítě na WWW.

Aplikace je koncipována architekturou klient–server.

Klientská část je naprogramovaná v JavaScriptu, DHTML, proto jí lze spustit v grafickém webovém prohlížeči. V editoru lze otevřít, navrhnout obecný graf, editovat jej, uložit ho na server a poté pokračovat v práci, nebo exportovat ve formátu StochPNTD[41] (vycházející z PNML, rozšířeného o stochastické a další grafické vlastnosti).

Serverová část je naprogramována v PHP. Spolupracuje s klientskou částí a je propojena s databází. Databáze slouží k autentizaci uživatelů, ukládání a načítání nakreslených grafů.

Největším problémem, jak se ukázalo, bylo vyřešit interaktivní kreslení v prohlížeči s použitím JavaScriptové a DHTML technologie. To jsem vyřešil kreslením pomocí výměny a změny obrázků. Dalším se stalo nedodržování webových standardů prohlížečů a nepřipravenost prohlížečů pro dynamické a interaktivní aplikace. Tyto problémy by se daly vyřešit použitím jiné technologie (např: Java, Flash), ale přinesli by závislost na jiných technologií a aplikace by se nestala volně přenosnou.

Aplikace je zprovozněna na webovém serveru (http://www2.stpnplay.com/grapheditor/) a lze ji na této adrese vyzkoušet.

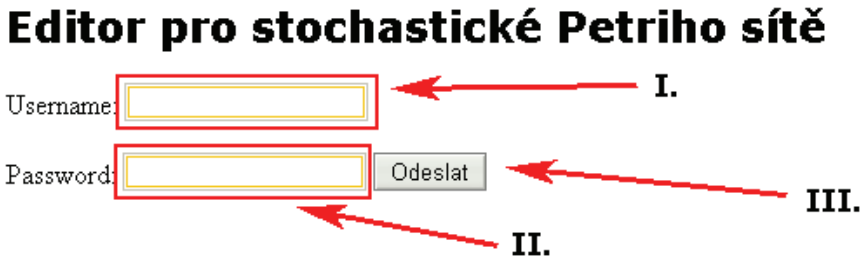
Editor lze použít pro nakreslení Petriho sítě, organizačního diagramu a dalších grafů.

[41] StochPNTD – XML formát pro popis stochastických Petriho sítí. Podrobněji v kapitole 2.13.

6 Přílohy

6.1 Příloha A – uživatelská příručka k editoru

6.1.1 Úvodní obrazovky



Obrázek 37 – Úvodní obrazovka

První úvodní obrazovka (Obrázek 37) slouží k přihlašování uživatelů. Kolonky *username*(I.) a *Password*(II.) slouží k zadání přihlašovacích údajů. Po vyplnění se tlačítkem *Odeslat*(III.) provede autentizace a uživatel se dostane na další obrazovku (Obrázek 38).

Obrázek 38 – Výběr možností editoru

V této obrazovce je na výběr několik možností.

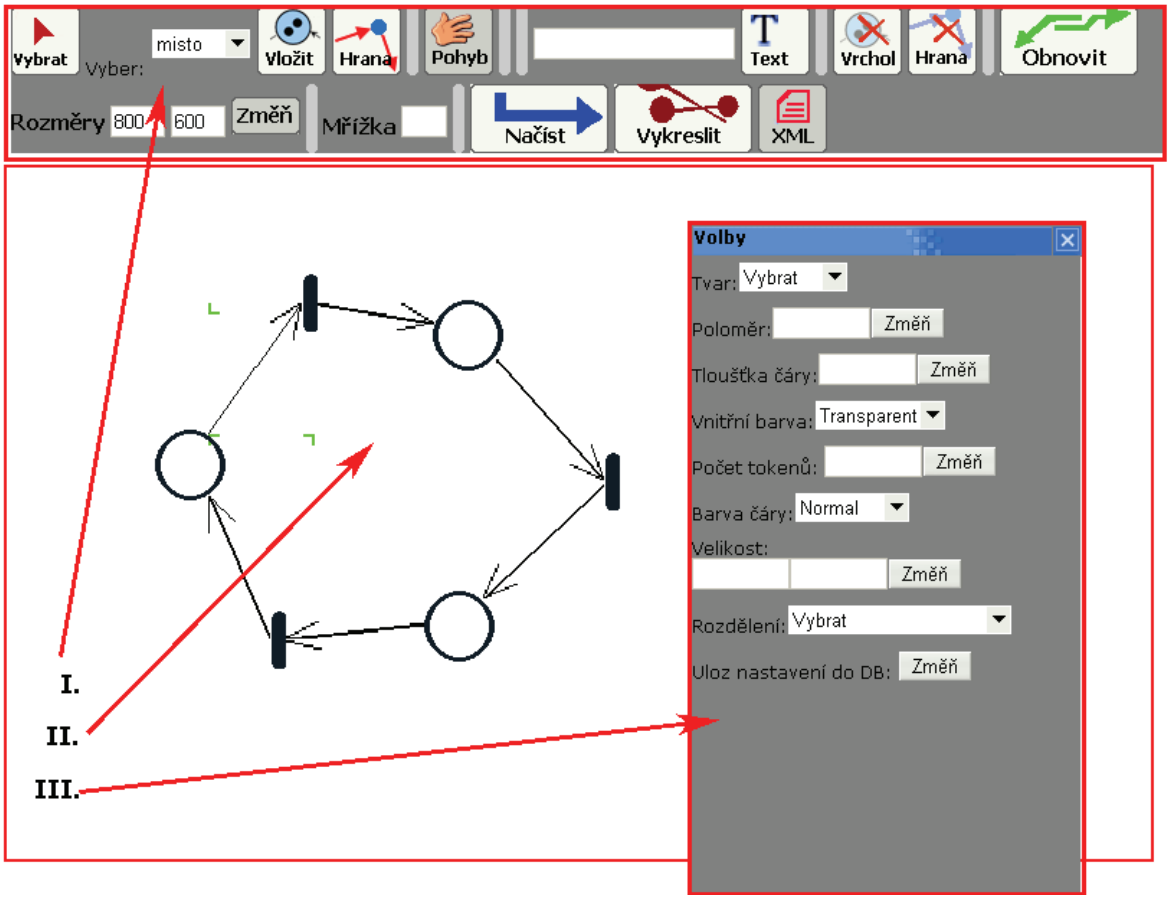
I. Uživatel může pokračovat a nahrát rozkreslený graf, který nakreslil v předešlém spuštění editoru.

II. Uživatel může začít kreslit nový graf. Při této volbě dojde ke zmazání rozkresleného grafu.

III. Vygenerovat stochPNTD rozkresleného grafu.

IV. Nahrát graf v stochPNTD (uložený na lokálním disku) na server a vykreslit ho.

6.1.2 Hlavní obrazovka a použití jednotlivých nástrojů



Obrázek 39 – Hlavní obrazovka editoru

Hlavní obrazovka se skládá ze tří částí.

I. Nástrojové menu

Tlačítko *Vybrat* – vybere kreslený objekt a zobrazí jeho vlastnosti.

Použití tlačítka: klikneme na tlačítko *Vybrat* a poté na nakreslený objekt.

Tlačítko *Vložit* – vloží vybraný prvek.

Použití tlačítka: vybereme kreslený objekt z výběrového prvku. Poté klikneme na tlačítko *Vložit* a poté klikneme do kreslící plochy. Na místo, kam klikneme se vloží nakreslený objekt.

Tlačítko *Hrana* – vloží hranu.

Použití tlačítka: klikneme na tlačítko *Hrana* a poté klikneme do kreslící plochy nejdříve na jeden objekt typu místo nebo hrana a poté na druhý. Tímto se vykreslí hrana mezi objekty.

Tlačítko *Pohyb* – pohybuje s objekty.

Použití tlačítka: klikneme na tlačítko *Pohyb* a poté klikneme do kreslící plochy na objekt typu místo nebo hrana. Poté odlehčíme

levé tlačítko na můžeme objektem pohybovat po kreslící ploše (spolu s navázanými hrany, má-li nějaké).

Tlačítko *Text* – vloží text.

Použití tlačítka: napíšeme do editovacího políčka text, poté klikneme na tlačítko *Text* a pak klikneme do kreslící plochy. Text se vykreslí na určenou pozici.

Tlačítko *Mazat vrchol* – smaže vrchol.

Použití tlačítka: klikneme na tlačítko Mazat vrchol a poté klikneme do kreslící plochy na vybraný objekt vrchol. Ten je po kliknutí smazán.

Tlačítko *Mazat hranu* – smaže hranu.

Použití tlačítka: klikneme na tlačítko *Mazat hrana* a poté klikneme do kreslící plochy hranu. Hrana je po kliknutí smazána.

Tlačítko *Obnovit* – obnoví graf (po pádu prohlížeče/zavření okna).

Použití tlačítka: Klikneme na tlačítko *Obnovit*. Graf se vykreslí do plochy. (pozn.: do kreslící plochy nesmí být kresleno).

Tlačítko *Změň* – nastaví velikost kreslící plochy.

Použití tlačítka: Napíšeme nové rozměry plochy do editovacích polí. Klikneme na tlačítko *Změň*. Okamžitě se změní velikost plochy.

Tlačítko *Mřížka* – nastaví velikost mřížky.

Použití tlačítka: napíšeme rozestupy mřížky do editovacích polí. Okamžitě se nastaví mřížka (pokud je pole nevyplněné, mřížka není dostupná).

Tlačítko *Načíst* – načte vložený soubor do prostředí.

Použití tlačítka: kliknutím se načte soubor a lze ho vykreslit.

Tlačítko *Vykreslit* – vykreslí načtený soubor.

Použití tlačítka: Po kliknutí se začne vykreslovat načtený soubor.

Tlačítko *XML* – vygeneruje XML soubor kresleného grafu.

Použití tlačítka: Klikneme na tlačítko a otevře se nové okno prohlížeče s vygenerovaným XML souborem grafu.

II. Kreslící plocha

Na kreslící plochu se vykreslují objekty po vybrání tlačítek vložit *vrchol*, *hranu*.

III. Volby a vlastnosti kreslených objektů

Po kliknutí na tlačítka *Vybrat* nebo *Pohyb* se vykreslí obsah voleb.

Volby obsahují vlastnosti objektů:

Tvar – umožňuje vybrat tvar objektu.

Poloměr – změní poloměr kružnice.

Tloušťka čáry – nastaví tloušťku čáry kružnice.

Vnitřní barva – nastaví vnitřní barvu objektu.

Počet tokenů – vykreslí do kružnice zadaný počet tokenů.

Barva čáry – zvolí barvu čáry objektu.

Velikost – nastaví velikost obdélníku u obdélníkového tvaru.

Rozdělení – Volba rozdělení umožňuje vybrat typ rozdělení a jeho parametry.

Uložit nastavení do DB – uloží parametry objektu do databáze.

6.2 Příloha B – obsah přiloženého CD

DP-TomášIvánek2006.doc DP-TomášIvánek2006.pdf	Diplomová práce ve formátu Word a PDF
Editor/	Zdrojové soubory aplikace

7 Literatura a reference

[1] Vector Markup Language (VML) [online] [cit. 2006–06–11]
URL: <http://www.w3.org/TR/NOTE–VML>

[2] VML editor [online] [cit. 2006–06–11]
URL: <http://www.dynamicdrive.com/dynamicindex11/editor.htm>

[3] Web Development, JavaScripts for Download [online] [cit. 2006–06–11]
URL: <http://www.walterzorn.com>

[4] Online Function Grapher, [online] [cit. 2006–06–11]
URL: <www.walterzorn.com/grapher/grapher_e.htm >

[5] Multimedia Systems and Applications – Image Quality Computation [online] [cit. 2006–06–11]
URL: <http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html>

[6] ImageMagick: Command–line Tools: Compare [online] [cit. 2006–06–11]
URL: <http://www.imagemagick.org/script/compare.php>

[7] ImageMagick: Convert, Edit, and Compose Images [online] [cit. 2006–06–11]
URL: <http://www.imagemagick.org/script/index.php>

[8] The Apache HTTP Server Project [online] [cit. 2006–06–11]
URL: <http://httpd.apache.org/>

[9] PHP Hypertext preprocessor, [online] [cit. 2006–06–11]
URL: <http://www.php.net/>

[10] PostgreSQL, [online] [cit. 2006–06–11]
URL: <http://www.postgresql.org/>

[11] World Wide Web Consortium, [online] [cit. 2006–06–11]
URL: <http://www.w3.org/>

[12] Mozilla Firefox, [online] [cit. 2006–06–11]
URL: <http://www.mozilla–europe.org/cs/products/firefox/>

[13] Opera Web Browser, [online] [cit. 2006–06–11]
URL: <http://www.opera.com/>

[14] Microsoft Internet Explorer, [online] [cit. 2006–06–11]
URL: <http://www.microsoft.com/windows/ie/default.mspx/>

[15] Wikipedie, otevřená encyklopedie, [online] [cit. 2006–06–11]
URL: <http://cs.wikipedia.org/wiki/>

[16] Root.cz – linux, open source a free software, [online] [cit. 2006–06–11]
URL: <http://www.root.cz>

[17] Document Object Model (DOM) Level 2 Core Specification, [online] [cit. 2006–06–11]
URL: <http://www.w3.org/TR/DOM–Level–2–Core/Overview.html#contents>

[18] Drawing lines in JavaScript, [online] [cit. 2006–06–11]
URL: <http://www.p01.org/articles/DHTML_techniques/Drawing_lines_in_JavaScript/>

[19] AJAX, [online] [cit. 2006–06–11]
URL: <http://php.vrana.cz/ajax.php/>

[20] Velikost nahrávaných souborů, [online] [cit. 2006–06–11]
URL: <http://php.vrana.cz/velikost–nahravanych–souboru.php>

[21] Kosek, Jiří. Vše o WWW, [online] [cit. 2006–06–11]
URL: <http://www.kosek.cz/>

[22] KOSEK, Jiří. PHP tvorba interaktivních internetových aplikací . 1. vyd. Praha: Grada, 1999. 492 s. ISBN 80–7169–373–1.

[23] The official specification for ECMAScript, [online] [cit. 2006–06–11]
URL: <http://www.ecma–international.org/publications/standards/Ecma–262.htm>

[24] Jak psát web o tvorbě internetových stránek, [online] [cit. 2006–06–11]
URL: <http://www.jakpsatweb.cz/>

[25] Generování obrázků v PHP, [online] [cit. 2006–06–11]
URL: <http://www.pcsvet.cz/art/article.php?id=2226>

[26] ZAKAS, C. Nicholas. Professional JavaScript for Web Developers. Indianapolis, Indiana: Wiley Publishing Inc., 2005. 646 s. ISBN–13: 978–0–7645–7908–0.

[27] Drawing antialised lines, [online] [cit. 2006–06–11]
URL: <http://www.kosmokrator.com/download/php/>

[28] Trang Manual, [online] [cit. 2006–06–11]
URL: <http://www.thaiopensource.com/relaxng/trang–manual.html>

[29] AJAX kalendář, [online] [cit. 2006–06–11]
URL: <http://cal.webkitchen.cz/>

[30 Basecamp, [online] [cit. 2006–06–11]
URL: <http://www.basecamphq.com/>

[31] Gmail, [online] [cit. 2006–06–11]
URL: <http://www.gmail.com/>

[32] The Web Word Procesor, [online] [cit. 2006–06–11]
URL: <http http://www2.writely.com/info/WritelyOverflowWelcome.htm>

[33] Free Web Office Suite, [online] [cit. 2006–06–11]
URL: <http://www.goffice.com/>

[34] Instant Messaging with Ajax, [online] [cit. 2006–06–11]
URL: < http://www.meebo.com/>

[35] Flickr – Photo Sparing, [online] [cit. 2006–06–11]
URL: <http://www.flickr.com/>

[36] Portál Seznam, [online] [cit. 2006–06–11]
URL: <http://www.seznam.cz/>

[37] Yui tetris, [online] [cit. 2006–06–11]
URL: <http://www.dustindiaz.com/basement/tetris.html>

[38] Simple Ajax Toolkit, [online] [cit. 2006–06–11]
URL: <http://www.modernmethod.com/sajax/>

[39] JavaScript Object Notation, [online] [cit. 2006–06–11]
URL: <http://www.json.org/>

[40] Hanzálek, Zdeněk. Petriho sítě a Grafcet. [online] [cit. 2006–06–11]
URL: <http://labe.felk.cvut.cz/~tkrajnik/sdu/data/K333/Hanz01.PN.automatizace.pdf/>

[41] HTML 4.01 Specification. [online] [cit. 2006–06–11]
URL: <http://www.w3.org/TR/REC–html40/>

[42] Ajax: A New Approach to Web Applications. [online] [cit. 2006–06–11]
URL: <http://www.adaptivepath.com/publications/essays/archives/000385.php>

[43] PNML Animator. [online] [cit. 2006–06–11]
URL: <http://www.petriweb.org/pnml2png2/>

[44] XSLT transformations on PNML. [online] [cit. 2006–06–11]
URL: <http://wwwis.win.tue.nl/~jmw/pnml/>

[45] BUBENÍK, František, PULTAR, Milan, PULTAROVÁ, Ivana. Matematické vzorce a metody. 1. vyd. Praha: Vydavatelsví ČVUT, 1997. 313 s. ISBN 80–01–01643–9.

[46] Šremer, Martin. Diplomová práce – Editor pro Stochastické Petriho sítě na WWW pomocí Java technologií. 2006. CVUT–FEL.

[47] The Web Standards Project – ACID2 test. [online] [cit. 2006–06–11]
URL: <http://webstandards.org/action/acid2/guide/>

[48] Software and Systems Engineering – High-level Petri Nets Part 2. [online]
[cit. 2006–06–11]
URL: <http://www.petrinets.info/docs/ISO-IEC15909-2.WD.V0.9.0.pdf>