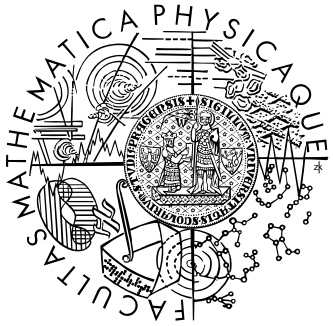
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Kozák

Petriho sítě

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: RNDr. Rudolf Kryl

Studijní program: Informatika, studijní obor Programování

2008

Chtěl bych poděkovat mému vedoucímu panu RNDr. Rudolfu Krylovi za jeho trpělivé vedení, za jeho čas a za jeho rady a postřehy, které v mnoha ohledech vylepšili tuto práci. Velmi si ho vážím. Dále bych chtěl poděkovat rodině, že mě celou dobu podporovala.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 6. 8. 2008 Michal Kozák

Obsah

Abstrakt

Název práce: Petriho sítě
Autor: Michal Kozák
Katedra (ústav): Kabinet software a výuky informatiky
Vedoucí bakalářské práce: RNDr. Rudolf Kryl
e-mail vedoucího: Rudolf.Kryl@mff.cuni.cz
Abstrakt: Cílem této práce je vytvořit grafický nástroj pro návrh a simulaci Petriho sítě. Pro praktické využití sítě si uživatel nadefinuje vlastní vzory, které umožní snazší návrh sítě. Tyto vzory lze opětovně použít při návrhu dalších sítí. Průběh simulace sítě lze sledovat a ladit pomocí podmíněných breakpointů. Pro řízení běhu simulace se využívá skriptovacího jazyka. Aplikace do skriptovacího jazyka zavádí rozhraní pro ovládání sítě. Uživatel si vytvoří procedury, na něž se dá v průběhu simulace odkazovat. Program je určen pro operační systém MS Windows.
Klíčová slova: Petriho síť, simulace, skript

Title: Petri Net
Author: Michal Kozák
Department: Department of software and computer science education
Supervisor: RNDr. Rudolf Kryl
Supervisor's e-mail: Rudolf.Kryl@mff.cuni.cz
Abstract: The aim of this work is to create a graphic tool for development and simulation of Petri nets. For practical and easier use of a Petri net the user defines a template for elements of the Petri net. These templates can be used in development of future Petri nets. The process of simulation can be watched and debugged via conditioned breakpoints. The flow of the simulation is controlled by scripts. The application implements script interface to give the user a way to control components of the Petri net. The user can write scripts procedures and can call them in the script.
Keywords: Petri net, simulation, script

1 Editační program Penta

Penta slouží pro návrh a modifikaci Petriho sítě. Uživatel si navrhne vzory, které mu usnadní a urychlí pozdější návrh sítě. Jednotlivé elementy Petriho sítě se vytváří podle předem navržených vzorů a přejímají jejich vlastnosti. Vzory se vytváří pomocí nástroje, který je součástí Penty. Vzory se dají uložit odděleně od sítě, aby se daly opakovaně použít při návrhu dalších sítí.

Uživatel se může kdykoliv během návrhu přepnout do simulačního režimu, který mu umožňuje otestovat a sledovat chování jím navržené sítě.

2 Motivační příklad.

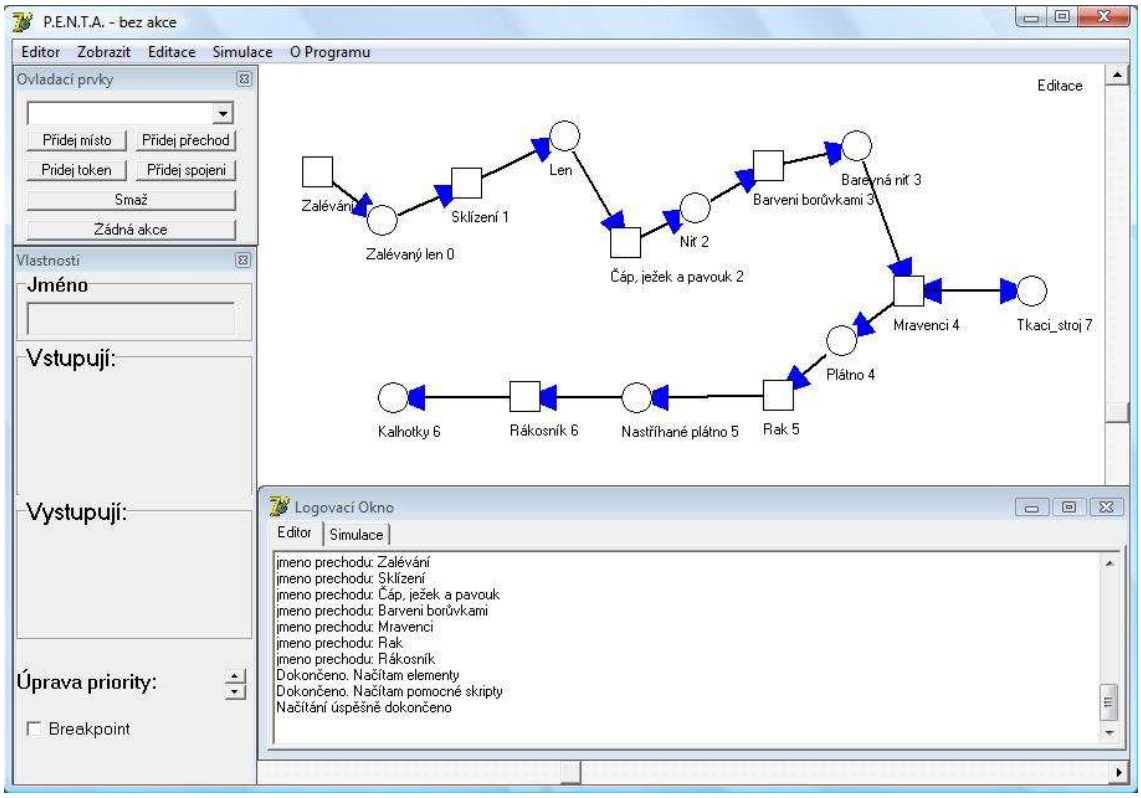
Než nadefinuji, co je to přesně Petriho síť, uvedu následující příklad. Každý asi zná pohádku o Krtečkovi. Jeden z jeho příběhů vypráví o jeho cestě za kalhotkami. Pro připomenutí napíšu nejprve krátké shrnutí obsahu:

Krteček si usmyslel, že chce nové kalhotky s kapsami. Po počátečním hledání zjistil, že na výrobu potřebuje len, který zpracuje na nit a z niti utká plátno. Plátno sestříhá a sešije a bude mít kalhotky. Samozřejmě tyto činnosti Krteček nikdy nedělal, avšak zvířátka mu pomohla. Aby krteček dostal len, musí tuto rostlinu nejprve zalévat. Když len vyrostl, tak ho Krteček donesl k čápovi, který ho rozlámal. Ježeček ho rozčesal a pavouk z něj upletl niť. Tuto niť Krteček obarvil borůvkami. Mravenci postavili tkací stroj, na kterém utkali plátno. Rak si Krtečka změřil a plátno rozstříhal. Nakonec Krteček donesl nastříhané plátno k rákosníčkovi, který mu ušil kalhotky.

V celé pohádce se objevuje základní téma. Krteček má meziprodukt a ten si nechá u někoho zpracovat. Například aby získal plátno, potřebuje niť a tkací stroj. Plátno nezíská, pokud bude mít jenom niť nebo tkací stroj bez nitě.

To je i základní myšlenka Petriho sítě. Žádná akce se nevygeneruje, dokud nebudou všechny prerekvizity splněny. Touto prerekvizitou může být cokoliv, v našem příkladě niť, plátno,… Tyto věci se obecně v Petriho sítích označují jako tokeny. Tokeny jsou uchovávány, dokud nejsou použity. Oblasti, kde jsou tokeny uchovávány, se označují jako „místa“. Místa sama o sobě nic nedělají, o veškerou práci a rozhodování v síti se starají tzv. „Přechody“ (angl. Transition).

Otevřete si příklad „Krteček“, popíši vám, jak v Pentě vypadají místa, tokeny a přechody. Soubor se jmenuje krtecek.xml.



Načetla se vám síť, která bude simulovat krtečkovu pouť za kalhotkami. V této síti vidíte kolečka a čtverečky napojené šipkami. Kolečko znázorňuje místo a čtvereček přechod. Šipka ukazuje, jakým směrem se v síti mohou pohybovat tokeny. Tokeny jsou znázorněny v Petriho síti jako malá kolečka u míst. V příkladu krtečka zatím žádné tokeny nevidíte. Uvidíte je, až spustíte za chvíli simulaci.

- 

Symbol značící přechod
- 

Symbol značící místo bez tokenů
- 

Symbol značící místo s dvěma tokeny
- 

Symbol značící místo s pěti tokeny

2.1.1 Simulace Petriho sítě

Po dvojím zmáčknutí tlačítka F5 se sít rozpohybuje. Místům se budou přidávat a odebírat tokeny v pravidelných intervalech. Ve stručnosti popíši, co se v síti od začátku simulace až po její konec děje.

Nejprve se třikrát zalije len. Místu „Zalévaný len“ se během 3 kol přidají 3 tokeny. Když je len dostatečně zalit, sklidí se a krteček získá len. Následuje zpracování lnu na nit a jeho obarvení borůvkami. Na řadě jsou mravenci. Mravenci postaví tkací stroj. Po dokončení stavby tohoto stroje utkají plátno. Stroj zůstane postavený a výrobní proces pokračuje. Nakonec je plátno nastříháno a sešito v kalhotky.

Může vás napadnou otázka, proč se nepokračovalo v zalévání, nebo proč mravenci nepostavili víc tkacích strojů? Toto chování je definováno v síti. Na následujících stranách se dozvíte, jak postupovat při návrhu Petriho sítě, jak číst jednotlivé vlastnosti a jak docílit takového chování sítě, které potřebujete.

3 Definice Petriho sítě a jejích prvků

V této kapitole je sepsán souhrn všech elementů Petriho sítě, jakou v této síti hrají roli a jaké mají vlastnosti.

Petriho síť není žádná novinka. Za dobu své existence, která se počítá od 60. let minulého století, vzniklo mnoho modifikací. Já vás nyní seznámím se základní podobou Petriho sítě a její modifikaci, se kterou pracuje tento editor – Penta.

3.1 Původní Petriho síť

Tato síť je orientovaný bipartitní graf, který obsahuje dva druhy uzlů – místa a přechody. Hrany mezi uzly spojují pouze místo s hranou, nebo hranu s místem. Nemůže nastat, aby hrana spojovala dvě místa nebo dva přechody.

Místa mohou obsahovat tokeny. Jejich počet není omezen.

Token je atomická informace. Nemá další vlastnosti.

Vstupní místo označujeme místo, ze kterého vede hrana do přechodu. Pokud vede hrana z přechodu do místa, označujeme toto místo jako výstupní.

Přechod je uschopněn, pokud všechna jeho vstupní místa obsahují aspoň jeden token.

Pokud je přechod uschopněn, může *vypálit*. Chování je nedeterministické, a proto může nastat v intervalu ihned až nekonečno.

Vypálení přechodu odebere všem jeho vstupním místům právě jeden token a přidá všem jeho výstupním místům právě jeden token. Zároveň se provede akce, kterou tento přechod představuje.

3.2 Modifikace Petriho sítě

- Jedna se opět o orientovaný bipartitní graf s dvěma typy vrcholů – přechody a místa. Pro hrany platí stejné podmínky jako v původní síti. Stejně se rozlišují i vstupní a výstupní místa.
- Modifikace však zavádí nový pojem – vzor přechodu, vzor místa a vzor tokenu. V původním návrhu sítě se vyskytoval pouze jeden vzor přechodů, míst a tokenů. V této modifikaci patří každý přechod, místo nebo token k jednomu vzoru. Počet vzorů je konečný. V rámci jednoho vzoru mají všechny instance tohoto vzoru stejné vlastnosti. Různé vzory mohou mít různé vlastnosti i různý počet vlastností.

- Každý konkrétní element v navržené Petriho síti je instancí některého vzoru.
- Poslední změna je, že ne všechny hrany spojující místa s přechody jsou povoleny. Povolena jsou pouze některá spojení vzorů míst a vzorů přechodů.
- Vypálení přechodů již není nedeterministické, ale je plně v režii skriptu přechodů. Umožnění využití skriptů je podle mého názoru velké plus a nedeterministické chování umožňuje simulovat pomocí generátoru pseudo-náhodných čísel.

3.2.1 Popis přechodů, míst a tokenů

Token – může obsahovat vlastnosti - například velikost, barvu apod. Každá instance vzoru tokenu pak má pro každou vlastnost hodnotu – tedy u velikosti například malý/střední/velký a u barvy zelený/modrý/červený.

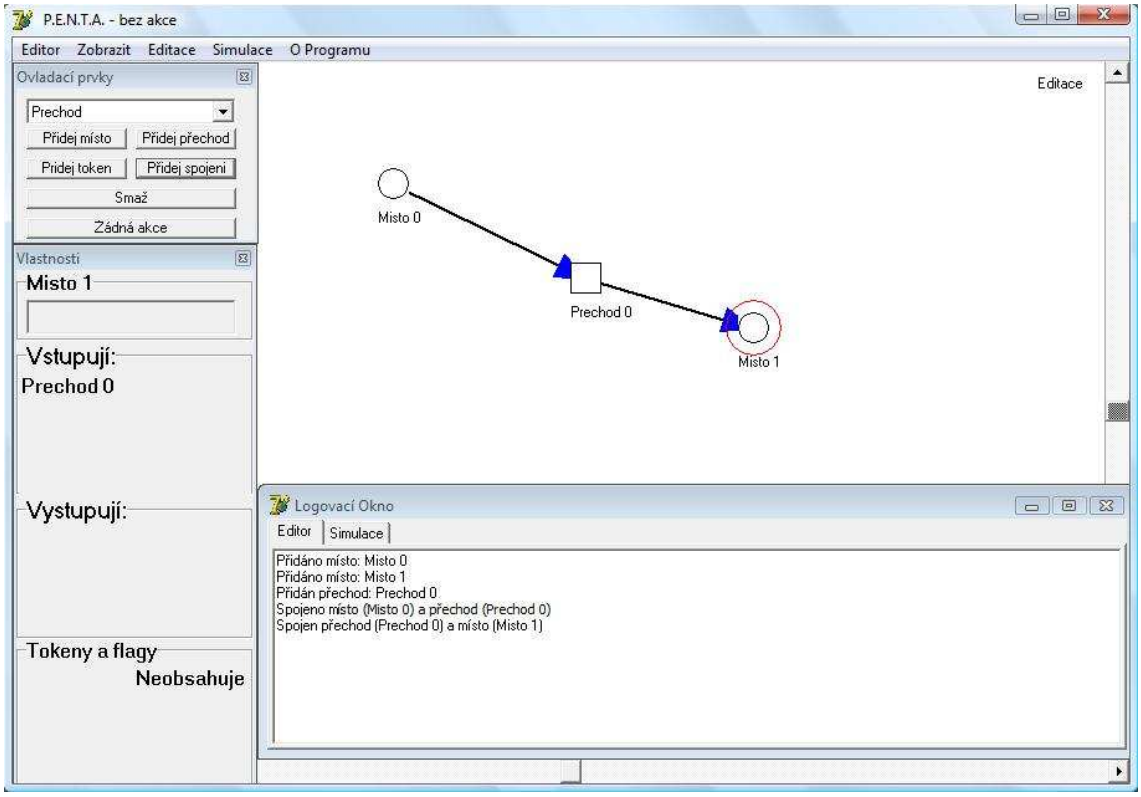
Místo – Každý vzor místa si určuje, jaké vzory tokenů může uchovávat. U instancí míst lze nastavovat příznaky – flagy. Seznam flagů je společný pro všechna místa bez rozdílu vzoru. Instance místa buď příznak obsahuje, nebo neobsahuje. Více se nerozlišuje.

Přechod – Každý vzor přechodu si definuje, jaké vzory míst mohou být jeho vstupní místa a jaké vzory výstupní místa. Dále se u každého vzoru přechodu definuje jeho funkce pro vypálení. Už se nerozlišuje, zda je přechod uschopněn – to rozhoduje tato funkce, která zároveň určí, jestli a jak tento přechod vypálí.

Oproti původní Petriho síti je tento návrh daleko členitější. U prvků se rozlišují jejich vzory, podle kterých mají své vlastnosti. Vzory přechodů mají vlastní funkci pro vypálení i uschopnění. Tato funkce se označuje jako přechodová funkce. Přechodová funkce může odebírat libovolné počty tokenů ze vstupních míst a přidávat libovolné počty tokenů výstupním místům.

4 Editor

Po spuštění aplikace se objeví okno jako na obr 2.



4.1.1 Vzhled - první pohled

Základní okno PENTA obsahuje editační plochu, nad kterou se vznášejí tři pomocná okna – „**Ovládací prvky**“, „Vlastnosti“ a „**Logovací okno**“. Všemi třemi okny se dá pohybovat nad plochou editoru nebo je lze skrýt a později opět zobrazit. Zobrazení skrytých oken se provede stisknutím příslušné volby v menu **Zobrazit**/.

Okno **Ovládací prvky** obsahuje funkce pro návrh a editaci Petriho sítě. Tímto oknem uživatel provádí volbu, jaké editační akce chce používat. Navolit lze přidávání jednotlivých prvků Petriho sítě, jejich mazání nebo jejich propojení. Akce přidávání využívá předem navržených vzorů. O vzorech si povíme níže v práci. Standardně se toto okno nachází v levém horním rohu editační plochy.

Okno **Vlastnosti** zobrazuje vlastnosti vybraného prvku Petriho sítě. Toto okno se nachází v levé části editační plochy pod oknem **Ovládací prvky**.

Veškeré provedené akce zanechají záznam v **okně pro logy**, které je po spuštění aplikace umístěno dole. Toto okno má dvě záložky – editační a simulační – v každé záložce je pak textové okno se záznamem akcí, které se provedly v daném režimu.

4.1.2 Pojmenování elementů PS na pracovní ploše

Pod každým místem či přechodem se nachází jeho pojmenování. Při vytvoření nového prvku se vygeneruje jméno následujícím způsobem: základ je jméno vzoru doplněný o mezeru a číslici. Místa a přechody se číslují zvlášť – pokud tedy bude například 15 elementů míst a 3 přechody, pak místa budou mít čísla 1 až 15 a přechody 1 až 3. Toto základní označení se zobrazí rovněž v případě najetí kurzorem myši nad daný prvek, což vyvolá zobrazení žlutého proužku s tímto označením.

Pro větší pohodlí je možné zavést vlastní pojmenování míst nebo přechodů. Označte element PS, který chcete přejmenovat. Přejděte do okna Vlastnosti. Zcela nahoře klikněte pravým tlačítkem myši na položku Jméno. Nyní můžete změnit jeho obsah.

4.1.3 Určení sousedských vztahů

Spojení mezi sousedy je znázorněno modrou šipkou. Směr šipky určuje, jakým směrem se mohou přemisťovat tokeny v síti. Pokud vede šipka ze čtverečku (přechodu) do kolečka (místa), jedná se o výstupní místo přechodu a obráceně, pro zmíněné místo je tento přechod vstupní soused. Seznam všech vstupních i výstupních sousedů je uveden v okně Vlastnosti. Dvojklikem na souseda se navolí na editační ploše a zobrazí se jeho vlastnosti v okně Vlastnosti.

4.1.4 Titulek PENTA

Titulek aplikace uživateli ukazuje, jaká aktuální akce je zvolena. Na obrázku není aktuálně zvolena žádná akce, proto je titulek „Editor – bez akce“. K vytváření Petriho sítě a jednotlivým akcím se vrátím později.

4.1.5 Rozlišení simulačního a editačního módu

Editor PENTA je jak editační tak i simulační nástroj. Pracuje ve dvou režimech: EDITAČNÍ a SIMULAČNÍ.

Editační režim slouží pro vlastní návrh vzorů pomocí okna Správce vzorů. V tomto režimu může uživatel pracovat se všemi editačními akcemi, které se nachází na okně Ovládací prvky. S přesným popisem akcí se seznámíte v následující kapitole. Editační režim umožňuje:

- Navrhovat a upravovat vzory
- Přidávat a mazat nová místa a přechody

- Přidávat, modifikovat nebo mazat tokeny
- Nastavovat flagy
- Propojovat přechody s místy

V simulačním režimu jsou editační akce zablokovány. Uživatel však může simulovat chování sítě a ladit ho. V simulačním módu může uživatel pouze měnit tokeny. Je povoleno je přidávat nebo mazat a upravovat hodnoty jejich vlastností.

Simulační režim umožňuje:

- Přidávat, mazat nebo upravovat tokeny
- Nastavovat flagy
- Simulovat chování navržené Petriho sítě.

V pravém horním rohu editační plochy je napsáno, v jakém módu se aplikace nachází - jestli v simulačním, nebo editačním. Přepínání mezi módy se uskutečňuje možnostmi v menu **Simulace/**.

4.2 Pomocná okna

Nad editační plochou Penty se mohou nacházet až 3 okna. Jedná se o již zmíněná okna Vlastnosti, Ovládací prvky a Logovací okno.

4.2.1 Ovládací prvky a editační akce

Jak již bylo výše uvedeno, na tomto pomocném okně jsou uvedeny základní editační akce. Akce se navolí kliknutím na příslušné tlačítko. Podle zvolené akce se upraví titulek Penta a popřípadě se zaplní combo box (na pomocném okně Ovládací prvky) vzory prvků PS.

Následuje výčet editačních akcí, jejich titulku, typu prvků v combo boxu a stručného popisu. Některé akce se po provedení zruší – čili po provedení akce se navolí automaticky akce „Žádná akce“. Některé akce naopak po provedení zůstávají v platnosti (přidávání míst)

Akce	Titulek	Combo box	Popis
Žádná akce	-	-	Neprovádí se žádná editační akce.
Přidej Místo	Přidání místa	Vzory míst	Po kliknutí na editační plochu se přidá na toto místo nový prvek PS odvozený od vybraného vzoru v combo boxu. Akce se neruší.

Přidej přechod	Přidání přechodu	Vzory přechodů	Po kliknutí na editační plochu se na daném místě vytvoří nový přechod podle vybraného vzoru. Akce se neruší.
Přidej Token	Přidání tokenů	Vzory tokenů	Po kliknutí na prvek místo se do tohoto místa pokusí přidat token podle navoleného vzoru. Akce se nemusí povést v závislosti na nastavení konkrétního vzoru místa, do kterého se pokouší tento token umístit. Akce se neruší.
Přidej spojení	Spojení	-	Klikněte na místo (resp. Přechod) a posléze na přechod (resp. Místo) Pokud to pravidla u vzoru přechodu umožní, vznikne napojení místa na přechod (resp. přechodu na místo) Akce se ruší.
Smaž	Mazání	-	Kliknutím na místo nebo přechod se tento prvek odstraní. Akce se neruší.

4.2.2 Okno Vlastnosti

Toto okno zobrazuje informace o vybraném místu nebo přechodu. Prvek PS se vybere kliknutím myši na jeho ikonu na editační ploše, pokud není právě navolená žádná editační akce. Vybraný prvek se zvýrazní červeným kolečkem.

V okně Vlastnosti se zobrazuje jméno vybraného prvku, dále jeho sousedé – vstupní a výstupní – a také další informace, v závislosti na tom, zda se jedná o místo nebo přechod. U míst se zobrazuje seznam uchovávaných tokenů a nastavených flagů. U přechodů se zobrazuje jejich priorita, kterou je možné u každého jednotlivého přechodu upravovat. Více o prioritách přechodů bude řečeno v kapitole Simulace.

jak bylo již uvedeno dříve, Jméno vybraného prvku lze měnit: Pravým tlačítkem myši nad starým označením prvku PS se zpřístupní editace, po ukončení editace jména se změna uloží.

Pro snazší nalezení sousedů je možné je vybrat přímo ze seznamu sousedů. Dvojklikem se tento soused vybere jak na editační ploše, tak v okně Vlastnosti.

U míst se zobrazuje seznam tokenů a flagů. Dvojklikem na konkrétní token se otevře okno, v němž je možné nastavovat hodnoty vlastností tohoto tokenu. Bližší informace o tomto okně se dozvíte v kapitole Návrh sítě – tokeny.

Seznam **flagů** se zobrazuje napravo od seznamu tokenů. Pokud místo nemá nastavené žádné flagy, je uvedeno pouze „Neobsahuje“, jinak je uveden výpis nastavených flagů. Dvojklik do tohoto seznamu zobrazí okno s přiřazením flagů. Bližší informace o tomto okně se dozvíte v kapitole 7.1.5 Nastavení flagů.

4.2.3 Logovací okno

Toto okno je místo, kde se zaznamenávají informace o provedených akcích. Filtrují se podle režimu, v jakém se Penta nachází – editační, simulační. Filtr se přepíná pomocí záložek v horní části tohoto okna. Většinu obsahu okna vyplňuje textové okno se záznamy. Nad tímto textovým oknem lze vyvolat pop-up menu a uložit obsah okna nebo jej smazat.

Třetí záložka „Stav sítě“ obsahuje výpis aktuálního stavu sítě. Tento výpis obsahuje seznam všech míst a obsah těchto míst. Obsahem místa zde myslím seznam nastavených flagů a seznam uchovávaných tokenů. Každý token má u sebe zapsány hodnoty vlastností.

5 Vzory - Správce vzorů

Petriho síť se skládá z elementů, které mají své vlastnosti a chování odvozeny od vzorů. Nyní si řekneme, co to vzory jsou, jak je vytvářet a upravovat.

Vzory spravuje okno Správce vzorů. K tomuto oknu se dostanete z hlavní nabídky: Editace/Správce vzorů…

Toto okno se skládá ze záložek. Každá záložka se věnuje jedné třídě vzorů nebo skriptů. Záložky jsou: Tokeny, Místa, Přechody, Flagy, Skripty a Breakpointy. Na každé záložce je na levé straně seznam zástupců, kteří pod danou skupinu spadají. Vybráním daného prvku ze seznamu se na pravé straně zobrazí podrobné vlastnosti.

Poznámka: Označením „vzor Petriho sítě“ myslím souhrnně vzory přechodů, míst a tokenů. V této kapitole používám označení vzor pro všechny prvky v seznamu Správce vzorů.

5.1.1 Mazání vzorů v seznamu

Pro smazání vzoru je nutné vybrat vzor ze seznamu a pod tímto seznamem zmáčknout tlačítko „Odebrat“. Nelze smazat první vzor v seznamu u tokenu, míst, nebo přechodů.

5.1.2 Vytvoření nového vzoru

Pro vzory Petriho sítě je nutné nejprve označit prvek ze seznamu, skripty a flagy tuto podmínku nemají. Po stisknutí tlačítka „Přidat“ je uživatel dotázán na nové jméno vzoru. Toto jméno musí být v dané skupině jedinečné. Následně je vytvořen nový vzor. Pokud se jedná o nový vzor Petriho sítě, použije se dříve označený vzor ze seznamu jako předloha a zkopírují se veškeré jeho vlastnosti. Poté jsou již na sobě oba vzory nezávislé.

5.1.3 Přejmenování vzoru Petriho sítě

U vzorů Petriho sítě je možné změnit jejich původní pojmenování. V horní části, vpravo od seznamu vzoru je textové pole s tlačítkem „Změnit jméno“. Přepsáním této hodnoty a stisknutím uvedeného tlačítka se jméno změní.

5.2 Záložka Flagy

Flag je značka, příznak pro místo. Jedinou informaci, kterou přenáší, je jeho přítomnost resp. absence. Každé místo může obsahovat libovolnou podmnožinu flagů z tohoto seznamu. Flagy nemají žádné další vlastnosti, pouze své jméno.

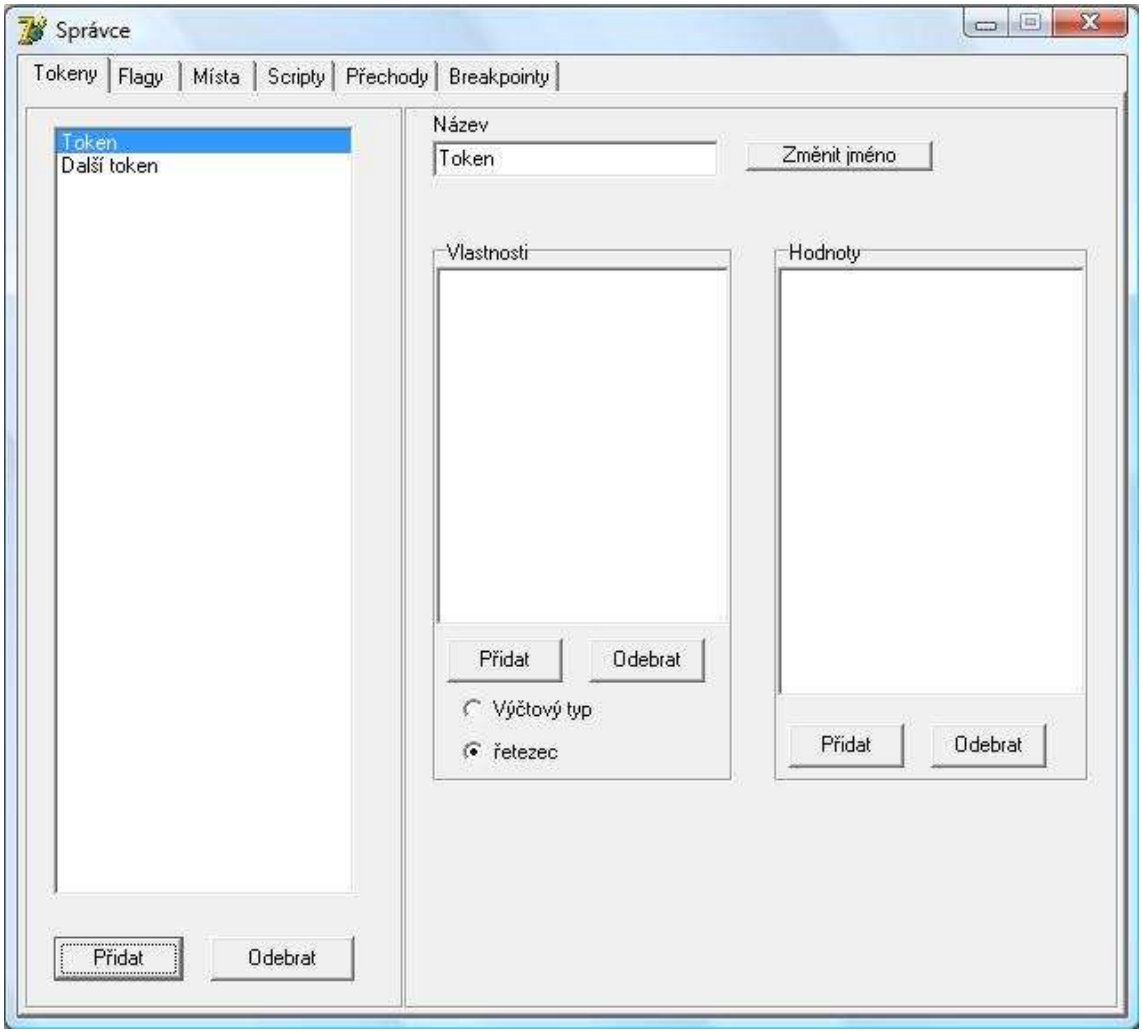
5.3 Záložka Tokeny

Každý vzor tokenu může (ale nemusí) obsahovat vlastnosti. Jednotlivé tokeny v PS, postavené na základě tohoto vzoru, pak mají stejné.

Penta podporuje dva typy vlastností: *Výčtové* a *řetězcové*.

Řetězcové vlastnosti uchovávají řetězce, které mohou být libovolně dlouhé.

Výčtové vlastnosti mají omezenou množinu legálních ohodnocení. Platné hodnoty si definuje samotný uživatel. Výčet vlastností je uveden v seznamu hodnot, který se nachází napravo od seznamu vlastností.



5.3.1 Čtení vlastností

Pro získání podrobnějších informací o nastavení vlastnosti vyberte danou vlastnost ze seznamu vlastností. Přepínač pod tímto seznamem se sám nastaví podle typu vlastnosti (výčtový typ/řetězec) a v pravém seznamu se vypíše seznam hodnot, pokud se jedná o výčtový typ.

5.3.2 Přidání nových vlastností tokenu

Pod seznamem „vlastnosti“ se nachází dvě tlačítka Přidat a Odebrat. Po stisknutí tlačítka „Přidat“ je uživatel dotázán na jméno nové vlastnosti. Jména vlastností musí být jedinečná pro daný vzor tokenu. Nová vlastnost bude přidána do všech již existujících tokenů vytvořených podle této třídy.

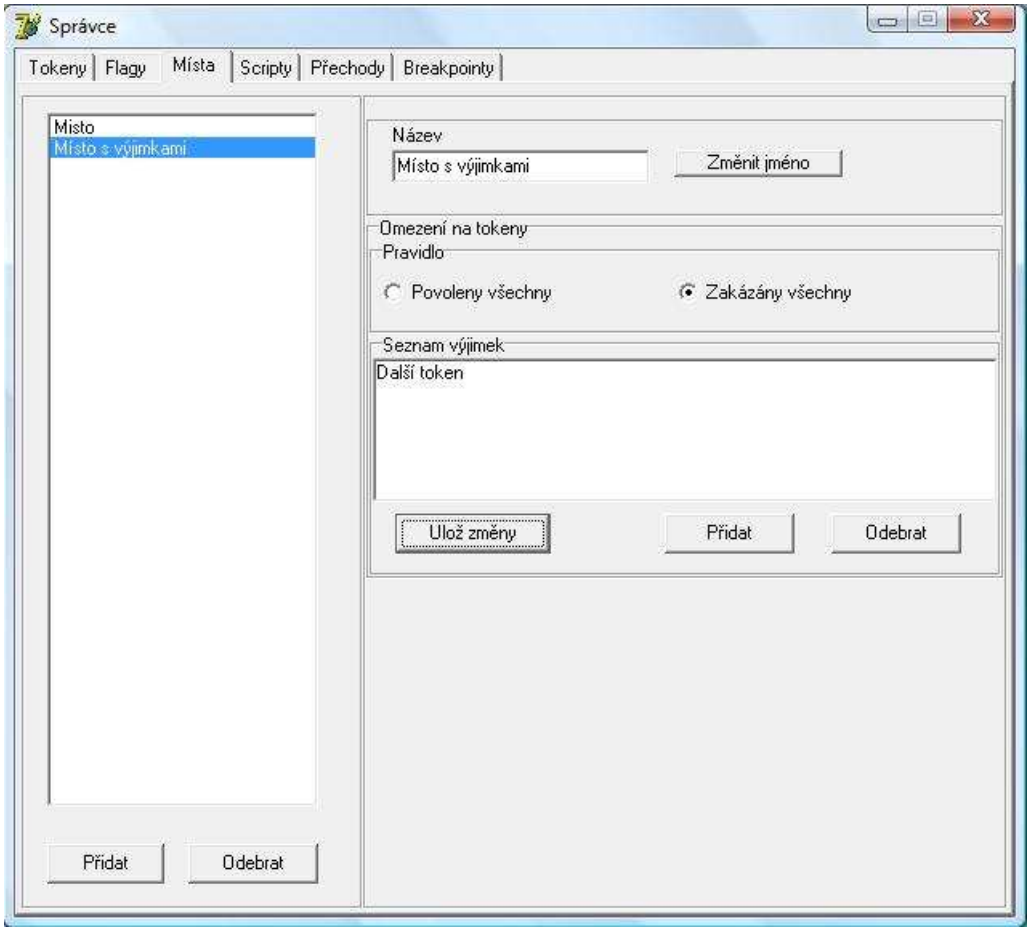
5.3.3 Mazání vlastností tokenu

Dalším tlačítkem, které se nachází pod seznamem Vlastnosti, je tlačítko Odebrat. Označte vlastnost v seznamu, kterou chcete smazat, a zmáčkněte toto tlačítko. Vlastnost bude odstraněna z tohoto seznamu i ze všech instancí tokenů, vytvořených podle toho vzoru Petriho sítě.

5.3.4 Změnění typu vlastnosti

Pro přepnutí vlastnosti z výčtového typu na řetězcový nebo opačně, vyberte danou vlastnost ze seznamu. Přepínač nastavte, aby byl vybrán vámi požadovaný typ. Změna se projeví u všech tokenů v Petriho síti. Pokud byla vlastnost původně vedena jako výčtový typ a měla již vytvořené hodnoty, pak se tyto hodnoty přepínáním neztratí. Po přepnutí na řetězcový typ se sice nezobrazí, ale opětovným přepnutím zpět se znovu v seznamu hodnot objeví.

5.4 Záložka Místa



Hlavním úkolem míst je uchovávání tokenů. U vzorů míst se proto definuje, jaké tokeny se smí v daném vzoru přechovávat.

5.4.1 Způsob určení povolených vzorů tokenů

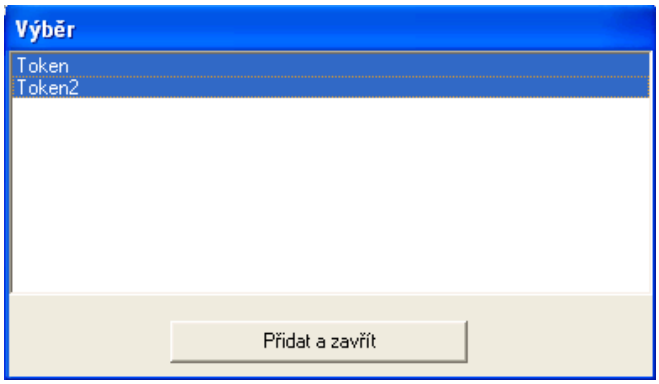
V rámci „Omezení na tokeny“ se nastavuje implicitní nastavení, zda povolit nebo naopak zakázat všechny vzory tokenů. Implicitní nastavení se omezuje výjimkami. Základní pravidlo je implementováno přepínačem. V seznamu pod tímto přepínačem jsou uvedeny výjimky.

Například: Přepínač je v poloze „Povoleny všechny“ a v seznamu je vzor „Token“. Pak tento vzor místa příjme všechny vzory tokenů kromě vzoru „Token“.

5.4.2 Přidání výjimek

Výjimky pro implicitní pravidlo se přidávají pomocí tlačítka „Přidat“. Po stisknutí se objeví okno se seznamem všech vzorů tokenů. V tomto okně označte

všechny vzory, které se mají použít jako výjimky. Výběr potvrďte stisknutím tlačítka „Přidat a zavřít“.



5.4.3 Odebrání výjimek

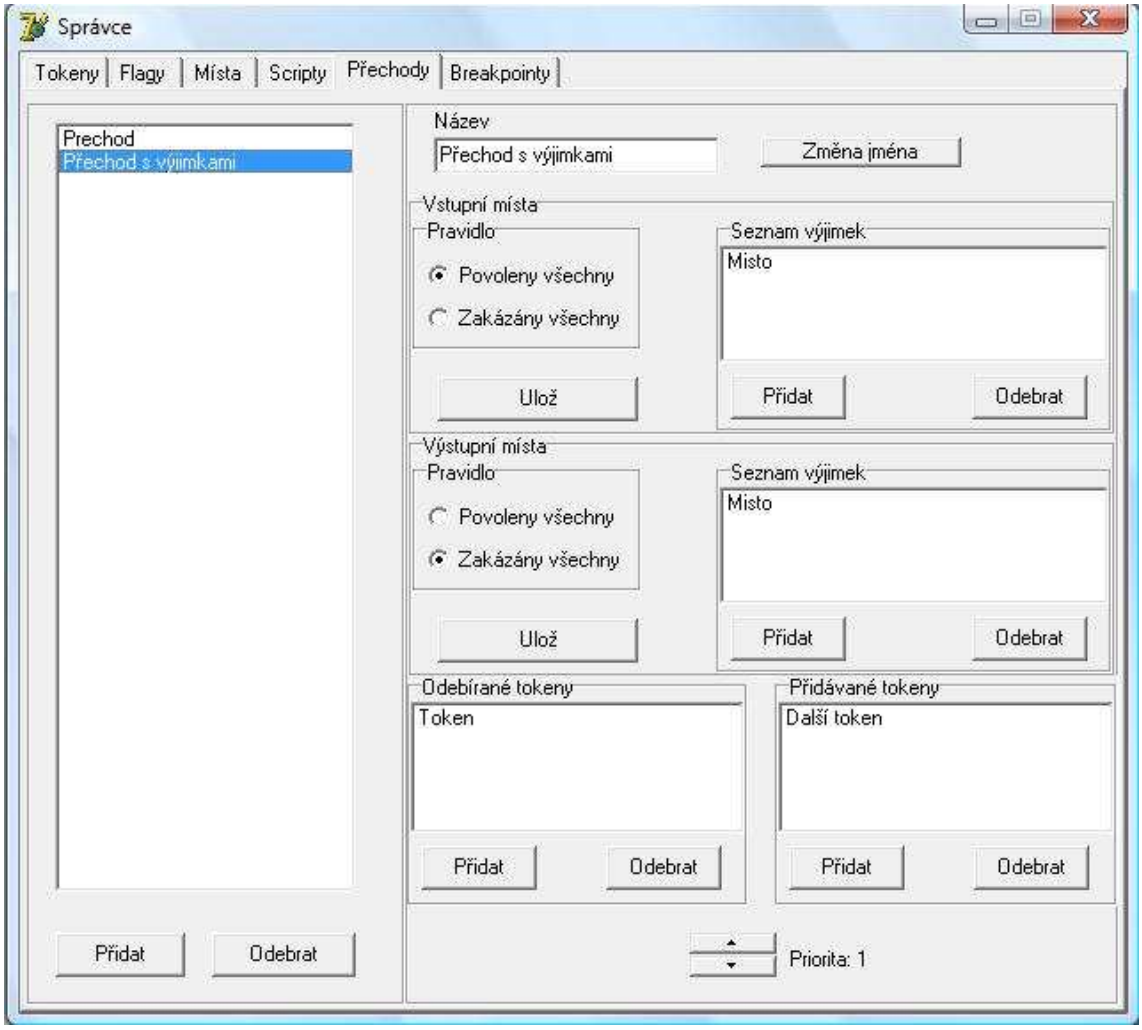
Označte výjimky ze seznamu (více výjimek lze označit podržením CTRL) a stiskněte „Odebrat“.

5.4.4 Uložení změn

Veškeré provedené změny je nutné uložit, jinak se neprojeví. Stisknutím tlačítka „Ulož změny“ se změna vzoru promítne do celého návrhu Petriho sítě. Každý token v Petriho síti, který nevyhovuje nově nastaveným pravidlům, bude z Petriho sítě vymazán.

5.5 Záložka Přechod

U vzorů přechodu se nastavuje, na jaká místa se může napojit. Pro usnadnění skriptování se dále uvádí takzvané „rozhraní tokenů“.



5.5.1 Určení pravidel pro napojení míst

Přechody mají dvoje omezení na místa, které se smí/nesmí napojit – vstupní a výstupní. Způsob nastavení pro obě množiny je však stejný.

Implicitní pravidlo povolení či zakázání napojení místa se určuje přepínačem v rámečku „Pravidlo“. Tento přepínač má dva stavy: Povoleny všechny/Zakázány všechny.

Platnost implicitního pravidla omezují výjimky. Jejich seznam je uveden napravo od přepínače.

Přidání výjimek pro pravidlo

Po stisknutí tlačítka „Přidat“ se objeví okno se seznamem všech vzorů míst, které ještě mezi výjimkami nejsou. Lze označit více vzorů naráz. Stiskem tlačítka „Přidat a zavřít“ se vzory míst přidají mezi výjimky.

5.5.2 Rozhraní tokenů

Pro každý vzor přechodů lze definovat dva seznamy vzorů tokenů – vstupní rozhraní a výstupní rozhraní. Tyto seznamy slouží uživateli při vytváření skriptů a jsou ze skriptu přístupné.

Přidání vzoru tokenu do seznamu probíhá obdobně jako přidání vzoru do výjimek. Tlačítkem „Přidat“ se objeví okno se seznamem všech vzorů tokenů.

Tlačítkem „Odebrat“ se ze seznamu odstraní označené tokeny.

5.6 Záložka Skript

Každý vzor přechodu má svoji přechodovou funkci. Tuto funkce zde označuji jako skript přechodu. Skript přechodu je v seznamu pojmenován podle svého vlastníka - přechodu. Skript přechodu se skládá z volání procedury skriptu a může také obsahovat volanou proceduru.

V seznamu jsou kromě skriptu přechodů uvedeny procedury a funkce, které nepatří k žádnému skriptu přechodu. Tyto procedury a funkce souhrnně označím jako P-procedury. V seznamu se rozliší předponou „P-„

Každý skript přechodu musí obsahovat platné volání procedury ať už vlastní, P-procedury nebo procedury jiného skriptu přechodu.

Vlastní skript se zavádí do paměti při startu simulačního režimu. Při psaní nových skriptů doporučuji, aby se uživatel pravidelně přepínal do simulačních režimů. Tím se snadněji odhalí chyby při překladu skriptů.

Každý skript se musí zapouzdřit do těla procedury, nebo zůstat prázdný.

5.6.1 Volání procedur

Volání procedur lze pouze nastavit u skriptu přechodů. Do textového okna „Volání skriptu“ zadejte jméno procedury skriptu. Obsah tohoto okna se použije pro volání skriptu při simulaci přechodu.

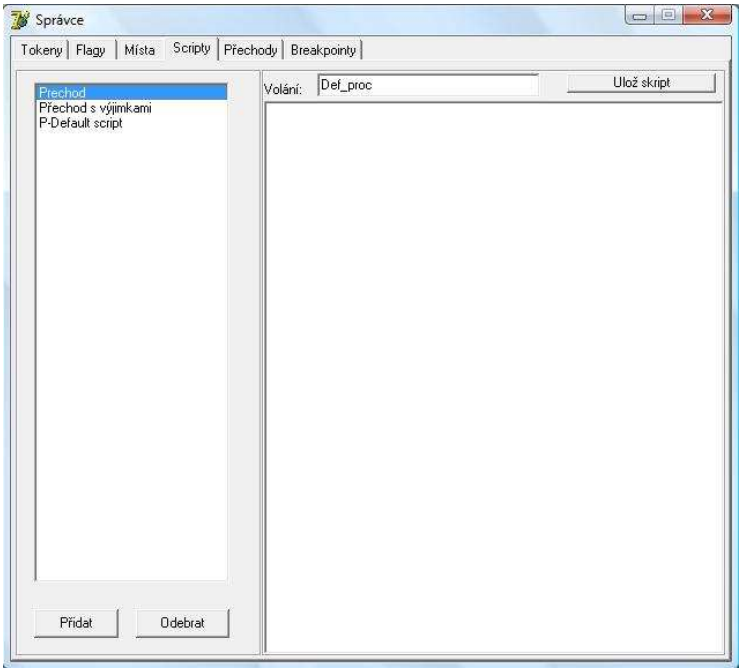
Volat procedury lze jen bez parametrů. Pro volání procedur s parametry je potřeba si vytvořit novou proceduru, která vytvoří tyto parametry a vykoná příslušné volání.

5.6.2 Defaultní skript

Při vytvoření nového projektu se automaticky generuje P-procedura, která řídí chování defaultního přechodu.

Tento skript se skládá ze dvou kroků:

- Kontroluje se každé vstupní místo, zda obsahuje alespoň jednoho zástupce od každého vzoru, který je uveden ve vstupním rozhraní tokenů.
- Pokud platí předešlý bod, provedou se následující akce. Každému vstupnímu místu se odebere nalezený zástupce pro každý vzor tokenu ze vstupního rozhraní. Každému výstupnímu místu se přidá jeden zástupce pro každý vzor tokenu z výstupního rozhraní.



5.7 Záložka Breakpointy

Penta umožňuje použití podmíněných breakpointů. Pro složitější rozhodování umožňuje Penta nadefinování uživatelských skriptů pomocí záložky Breakpointy.

Prostory skriptů pro breakpointy a skriptů pro přechody jsou oddělené. Volání pomocných procedur nebo přechodů ze skriptů breakpointů proto není možné.

Pro tyto skripty platí stejná pravidla jako pro skripty přechodů.

6 Návrh Petriho sítě

Nyní víme, jak navrhnout vzory. Za pomoci editačních akcí se seznámíme, jak navrhnout novou Petriho síť.

6.1.1 Návrh nosné struktury Petriho sítě

Pod pojmem nosná struktura jsou zde myšleny přechody a místa - tyto dvě množiny prvků vytváří základní infrastrukturu Petriho sítě.

6.1.2 Přidání míst a přechodů

Zobrazte okno s ovládacími prvky. Klikněte, vyberte akci „Přidat místo“ a ze seznamu vzorů vyberte vzor. Klikněte na editační plochu. Na tomto místě se vytvoří nový prvek PS – místo. Toto nově vytvořené místo ponese označení X 1, kde X je jméno vzoru. Pokud budete pokračovat v klikání na editační plochu, vytvoříte další místa - X 2, X 3, atd. Přechody se vytváří stejně jako místa – navolí se akce, „Přidat přechod“, zvolí se vzor a kliknutím na místo na editační ploše se přidá nový přechod. Přechody se přidávají do doby, dokud je navolená akce na přidání přechodů.

Po vytvoření nového místa nebo i přechodu se toto nové místo vybere a zobrazí se jeho vlastnosti.

Pro každý přidaný prvek se zapíše záznam do logovacího okna.

6.1.3 Propojení elementů PS

Před popisem navazování míst a přechodu připomínám, že ne všechny přechody je možné spojit s libovolným místem. Vzory přechodů mají nadefinována omezení pro místa, určující, se kterými vzory míst se mohou spojit.

Vyberte akci „Spojit místa a přechody“. Pořadí kliknutí na místo a přechod určí, jakým směrem budou moci téct tokeny. Po provedení napojení se akce dokončí a pro další napojení je nutné ji navolit znovu. Pokud tedy chceme, aby směr tokenů vedl z přechodu X do místa Y, bude sled operací následující: Zvolíme akci, klikneme na přechod X, klikneme na místo Y.

Pokud chcete, aby tokeny mohly v síti téct oběma směry, musíte vytvořit napojení v obou směrech. Sled akci s použitím předchozího příkladu: Zvolíme akci, klikneme na přechod X, klikneme na místo Y, zvolíme znovu akci propojení, klikneme na místo Y a nakonec na přechod X.

6.1.4 Přidání a nastavení tokenů

Opět upozorňuji, že vzory míst mají pravidla určující, které vzory tokenů mohou tato místa nést.

Zvolte akci „Přidej token“ a následně vyberte vzor pro nové tokeny. Každé kliknutí na místo na editační ploše přidá tomuto místu jeden token. Není omezeno, kolik tokenů může místo nést. Na editační ploše se počet nesených tokenů zobrazuje jako menší kolečka. Od 5 tokenů se ukazuje číslo značící jejich počet.

6.1.5 Editace hodnot tokenů

Okno Editor tokenu se vyvolá dvojklikem na jméno tokenu v seznamu tokenů (v okně vlastnosti)

Toto okno zobrazuje jméno tokenu, tlačítko pro jeho smazání a dále seznam všech vlastností a jejich ohodnocení.

Vlastnosti výčtového typu jsou reprezentovány combo boxem s vybraným ohodnocením. Vlastnosti řetězcového typu zobrazují svoji hodnotu v textovém poli.

Po zavření tohoto okna se změny uloží.

6.1.6 Nastavení flagů

Napravo od seznamu tokenů se nachází seznam flagů. Pokud je tento seznam prázdný, je místo tohoto seznamu uvedeno „Neobsahuje“. Dvojklikem do tohoto seznamu zobrazíte okno Nastavení flagů.

Na tomto okně jsou dva seznamy – „Obsahuje“ a „Seznam zbylých“. Mezi těmito dvěma seznamy lze přesouvat flagy. Seznam „Obsahuje“ obsahuje nastavené flagy pro dané místo.

6.2 Manipulace s projektem

Projekt představuje veškeré informace potřebné pro replikaci Petriho sítě – navržené vzory, skripty, polohu elementů Petriho sítě a přiřazené tokeny a flagy.

Veškeré informace projektu jsou uloženy v jednom souboru. Pro přenos projektu na jiný stroj tedy postačí přesunout právě tento jediný soubor s uloženým projektem. Další soubory nejsou nutné. Data v souboru jsou zapsána v XML v podobě, která by měla být čitelná i pro člověka. S přesnou podobou XML se seznámíte v příloze.

V menu Editor/ jsou na výběr tyto možnosti:

6.2.1 Nová síť

Smaže předešlou síť, její vzory a začne úplně nový projekt.

6.2.2 Uložit síť (CTRL + S)

Vyvolá dialog pro uložení. Uloží současnou síť i všechny navržené vzory, skripty,… do zvoleného souboru.

6.2.3 Uložit pouze vzory

Vyvolá dialog pro uložení souboru. Uloží jen vzory a skripty. Tato nabídka je vhodná pro pozdější využití hotových vzorů v novém projektu.

6.2.4 Nahrát síť (CTRL + L)

Vyvolá dialog pro otevření projektu. Načte celou PS, tj. vzory, skripty i jednotlivé prvky PS.

6.3 Stavy Petriho sítě

Stav zde myslím ohodnocení Petriho sítě (PS) – nastavené flagy, přiřazené tokeny a hodnoty tokenů. Při simulaci nebo i návrhu Petriho sítě se uživatel může přepínat mezi několika uloženými stavy aktuální PS. Načtení stavu Petriho sítě nastaví místům uložené tokeny a nastavení flagů. Všechny stavy jsou součástí projektu a přenáší se spolu s ním.

Správa stavů se vykonává pomocí okna „Správce stavů“. Toto okno se zobrazí z menu „Editace/Správce stavů…“. Na tomto okně je seznam všech stavů a tlačítka pro nahrání, vymazání a uložení současného stavu Petriho sítě.

6.4 Simulace

Simulace slouží pro otestování návrhu konkrétní Petriho sítě nebo její součásti. Nyní se tedy zaměříme na obsah menu Simulace, v němž jsou veškeré řídící prvky pro ovládání chodu simulace.

6.4.1 Přechod do simulačního módu

Přechod do simulačního módu se uskuteční stisknutím klávesy F5 nebo pomocí menu „Simulace/Mód simulace“. To, že se program nachází v simulačním režimu, si můžeme snadno ověřit tím, že je v pravém horním rohu editační plochy napsáno“ „Simulace“ (místo „Editace“). Logovací okno (pokud je zobrazené) se automaticky přepne na záložku pro simulaci.

Při přechodu do simulačního režimu se do paměti zavádí veškeré skripty. Seřazeny jsou podle priorit přechodů. Při zavádění skriptů se vypisují všechny překladové chyby těchto skriptů. Z tohoto důvodu se doporučuje průběžně při psaní skriptů přecházet do simulačního režimu a testovat je na chyby. Pokud se při zavádění skriptů nějaká chyba vyskytne, přepne se PENTA zpět do editačního režimu.

6.4.2 Ukončení simulačního režimu, přechod do editačního režimu

Přechod do editačního režimu se uskuteční stisknutím klávesy F10 nebo v menu pomocí Simulace/ukončit. Petriho síť se navrátí do stavu, v jakém se nacházela před zahájením simulace. Obnoví rozložení flagů i tokenů a jejich hodnot. Logovací okno se přepne zpět na záložku editace. Petriho síť lze uložit i v simulačním režimu.

6.4.3 Chyby při simulaci

Při přechodu do simulačního režimu nebo při samotné simulaci může v Petriho síti dojít k mnoha chybám. Pokud k nějaké chybě dojde, simulace se zastaví. Zatímco při chybě u přechodu do simulačního režimu, se PENTA vrátí do editačního režimu, při běhové chybě se pouze zastaví běh simulace.

Při výskytu chyby se vždy zapíše do logovacího okna řádek, příčina chyby, její původce a krátký popis této chyby. Například:

```
Chyba Scriptu(řádek 2):Syntax error
Chyba při zavádění skriptu p    echodu. P    echod Prechod
```

V příkladu nastala chyba na 2. řádku u skriptu přechodu „Prechod“ s popisem „Syntax error“.

PENTA neumožňuje krokovat kód skriptů.

6.5 Ovládání běhu simulace

Po přechodu do simulačního režimu je simulace připravena, avšak neběží. Je několik způsobů, jak uvést simulaci do pohybu – krokovat nebo ji nechat volně běžet. Dříve než vás seznámím s detaily jednotlivých operací, objasním jeden pojem:

Kolo simulace – Petriho síť se vyhodnocuje ve „skocích“. V jednom skoku se vyhodnotí všechny přechody v síti. Tento „skok“ označuji v simulačním režimu jako kolo simulace. V tomto kole simulace se provede skript všech přechodů. Pořadí vyhodnocovaných přechodů se během simulace nemění a je určeno jejich prioritou.

Jsou 3 možné druhy běhu simulace:

6.5.1 Běh bez přerušení

Tento typ se spustí stisknutím tlačítka F5 nebo se v nabídce menu vybere: Simulace/Spustit. V pravidelných intervalech se provádí kola simulace, přičemž simulace sama se nezastaví.

6.5.2 Simulace jednoho kola

Spustí se stisknutím tlačítka F8 nebo v menu Simulace/krok po kole. Simulace provede pouze jedno kolo a poté se pozastaví. Pokud bylo současné kolo simulace již částečně provedeno (například simulací po přechodu, viz níže), toto kolo se dokončí. Až při dalším kroku se provede simulace celého kola.

6.5.3 Simulace po přechodu

Spustí se stisknutím tlačítka F7 nebo v menu Simulace/krok. Simuluje se pouze jeden přechod a poté se simulace pozastaví. Aktuálně simulovaný přechod se zvýrazní červeným kolečkem.

6.5.4 Pozastavení běhu simulace

Běžící simulaci lze pozastavit stisknutím tlačítka F9 nebo v menu Simulace/pozastavit. Tímto způsobem však nedojde k jejímu ukončení, takže v ní lze následně pokračovat. Simulace se také pozastaví, pokud při vykonávání skriptu dojde

k chybě. Další způsob pozastavení běhu simulace je pomocí breakpointů, o kterých se dozvíte níže.

6.6 Breakpointy

Breakpointy slouží k zastavení běhu simulace. Nastavují se u přechodů a kontrolují se dřív, než se provede skript daného přechodu. Každý přechod může mít nanejvýš jeden breakpoint. V Pentě existují dva druhy breakpointů – podmíněné a nepodmíněné.

6.6.1 Nepodmíněný breakpoint

Nepodmíněný breakpoint zastaví běh simulace pokaždé, když program dojde k přechodu, u něhož je nastaven.

6.6.2 Podmíněný breakpoint

Tento breakpoint nemusí zastavovat běh simulace pokaždé, když program dojde k přechodu, u něhož je nastaven. Rozhodnutí, zda má tento breakpoint zastavit běh simulace, či nikoliv, je závislé na jeho podmínce. Podmínka je výraz, který vrací booleovskou hodnotu a je pomocí něj možné volat procedury a funkce nadefinované ve Správci vzorů u skriptů breakpointů.

6.6.3 Nastavení breakpointů

Breakpointy se nastavují u přechodu v okně Vlastnosti. Úplně dole v tomto okně se nachází zaškrtávací políčko Breakpoint. Pokud je zaškrtnuto, je breakpoint aktivován a kolem tohoto přechodu se zobrazí zelené kolečko. Tímto způsobem nelze nastavovat podmínku pro podmíněné breakpointy.

Přehled všech přechodů a jejich breakpointů můžete získat v menu Simulace/Breakpointy… V tomto okně je po levé straně seznam všech přechodů. Po pravé straně jsou přepínače určující, zda je breakpoint aktivován nebo deaktivován. Pod těmito přepínači je pole pro podmínku breakpointu – „Vyhodnocovaný výraz“. Po napsání tohoto výrazu je nutné zmáčknout tlačítko „Uložit výraz“, aby se trvale uložil.

V tomto okně se podmíněné a nepodmíněné breakpointy rozlišují pouze tím, zda mají nějaký výraz k vyhodnocení. Nepodmíněné breakpointy žádný výraz nemají.

6.7 Priority přechodů

Přechody se nevyhodnocují v náhodném pořadí. Jejich pořadí je určeno jejich prioritou. Při přechodu do simulačního režimu se všechny přechody seřadí dle priorit, tj. od největší po nejmenší a dle tohoto řazení se vyhodnocují jejich skripty. Vzájemné pořadí skriptů se stejnou prioritou není určeno.

Výsledná priorita přechodu PS je určena prioritou vzoru. Tato hodnota je ještě upravena o hodnotu, kterou uživatel může nastavit u každého jednotlivého přechodu v rozmezí -10 až +10. (Defaultně je 0).

6.8 Úprava Petriho sítě v simulačním režimu

Jak již bylo zmíněno dříve, v simulačním režimu jsou editační možnosti značně omezené. Ze všech editačních akcí je povolena pouze manipulace s tokeny a flagy. V simulačním režimu nelze přidávat nebo mazat místa, přechody ani jejich spojení.

Přidávání tokenu a nastavování jejich vlastností se provádí stejně jako v editačním režimu. Viz sekce Návrh sítě – Tokeny.

6.9 Skriptování

Důležitou součástí PENTY je skriptování. PENTA sama skripty neinterpretuje, nýbrž využívá služeb Microsoft Windows Script Host. Tato služba je součástí všech Windows od verze Windows 98SE.

Standardně jsou podporovány jazyky VBscript a Javascript. Další mohou být do Windows nainstalovány - PENTA umožňuje používat libovolný jazyk, který je ve Windows Script Host podporován.

6.9.1 Výběr skriptovacího jazyka

Jako výchozí jazyk programu je nastaven VBscript. Pro použití jiného skriptovacího jazyku je nutné nastavit konfigurační soubor. Nastavení viz příloha.

6.10 Základy VB skriptu

Tento jazyk se nejčastěji používá při skriptování webových stránek, může však být použit i nezávisle. Jazyk je odvozen od programovacího jazyka Visual Basic a my si nyní ukážeme základy jeho syntaxe.

6.10.1 Vlastnosti VBscriptu

- Case insensitive (nerozlišuje velikost písmen)
- Není nutné deklarovat proměnné, pokud překladač narazí na neznámý identifikátor, automaticky ho bere jako proměnnou.
- Konce řádku značí konec příkazu
- Operátor přiřazení je „=“
- Identifikátor musí začínat písmenem
- Řetězce se spojují (konkatenace) přes operátor &
- Komentář je uvozen znakem ‘

6.10.2 Proměnné

Proměnné se deklarují pomocí klíčového slova *dim* nebo jejich prvním výskytem.

```
Dim promenna
```

Nebo prvním výskytem

```
Promenna=1
```

Po prvním přiřazení do proměnné se už provádí její typová kontrola.

Pole

Deklarace **tříprvkové** pole:

```
Dim pole(2)
```

Na rozdíl od jiných jazyků se v deklaraci počítá jeho velikost včetně nuly. Na prvky pole se přistupuje již normálně (index počítán od nuly)

```
Pole(0)=“hodnota 1“
Pole(1)=“hodnota 2“
Pole(2)=“hodnota 3“
```

6.10.3 Podprogramy

Pokud má definice podprogramu 0 argumentů, může se deklarovat buď s prázdnými závorkami, nebo bez nich.

Definice procedur

```
Sub identifikátor ([parametr]*)
  Tělo
  Exit sub ‘předčasné ukončení procedury
End sub
```

Definice funkce:

```
Function identifikátor ([parametr]*)
  Tělo
  Identifikátor=návratová_hodnota
  Exit function ‘předčasné ukončení funkce
End function
```

Volání podprogramu:

```
Identifikátor parametr1, parametr2,…
```

6.10.4 Větvení programu

If-then struktura:

```
If podmínka then Příkaz ‘vše musí být na 1 řádku
```

Nebo

```
If podmínka then
  Příkazy
End if
```

If-then-elseif-else struktura:

```
If podmínka then
  Příkaz
ElseIF podmínka then
  Příkaz
Else
  Příkaz
End if
```

Select case

```
Select case promenna
    Case „hodnota“
        Příkazy
    Case „jina hodnota“
        Příkazy
    Case Else
        Příkazy
End select
```

6.10.5 Cykly

For cyklus základní

```
For i=1 to 10
        kód
Next
```

For cyklus s definovaným inkrementem (STEP)

```
For i=2 To 10 Step 2
  Kód
Next
```

Hodnota po Step může být i záporná. Pozor na zacyklení! For cyklus se dá ukončit klíčovým výrazem **exit for.**

For Each...Next

Tento cyklus se iteruje pro každou položku kolekce nebo pole.

```
dim auto(2)
auto (0)="Volvo"
auto (1)="Saab"
auto (2)="BMW"
For Each x in auto
        MsgBox x
Next
```

DO..Loop

```
Do While i>10
  Kód
Loop
```

Nebo s vyhodnocením podmínky na konci:

```
Do
  Kód
Loop While i>10
```

Místo klíčového slova WHILE se dá použít ještě klíčové slovo UNTIL. WHILE – dokud platí podmínka, UNTIL – dokud nenastane (např.: until i=10 se bude provádět, dokud bude **i** různé od 10).

Opuštění Do..Loop smyčky: **EXIT DO**

6.11 Skriptové rozhraní PENTA

Každý skript, který je vykonáván, má přístup k těmto rozhraním, které mu umožňují ovládat běh simulace: Vstupni, vystupni, aplikace, prehled, rozhrani_vstup a rozhrani_vystup.

6.11.1 Rozhraní vstupni a vystupni

Vstupni, Vystupni je pole vstupních (resp. výstupních) míst. Má tyto metody a vlastnosti:

Pocet vrátí počet míst v poli.

Misto je pole jednotlivých míst. Místo má rozhraní *MISTO*.

6.11.2 Rozhraní Misto

Misto zastupuje konkrétní místo v Petriho síti. Obsahuje tokeny a flagy.

Pocet vrátí počet uchovaných tokenů.

Token pole, které umožňuje přístup k jednotlivým tokenům. Prvky tohoto pole mají rozhraní *TOKEN*

Pridej(id_třídy_tokenu). Přidá nový token tomuto místu. Tento nový token se stane viditelným až v dalším kole, v současném kole se s ním nemůže nakládat jinak, než ze skriptu, který jej vytvořil. Tato metoda vrací rozhraní TOKEN.

```
‘všem vstupním místům přidá 1 token a vytiskne třídu tohoto tokenu
For i=0 to vstupni.pocet -1
  Set pridany_token = vstupni.misto(i).pridej(0)
  Aplikace.pridej_log pridany_token.trida
next
```

V tomto kusu kódu se přidá všem vstupním místům token a vytiskne se jeho třída (která je 0).

Odeber(index_tokenu) - Odebere token na daném místě pole. Pole je pak neaktuální. Může dojít k přeházení tokenů. Proto by se při sekvenčním průchodu pole mělo začít procházet od začátku. Odebrání tokenu se z pohledu sítě projeví okamžitě (žádný jiný přechod tento token již nemůže odebrat).

Trida vrací číslo třídy tohoto místa.

Ziskej_flag(index_flagu) - vrátí hodnotu 0 (flag není nastaven), nebo 1(flag je nastaven)

Nastav_flag (index_flagu, hodnota). Nastaví hodnotu flagu u tohoto místa, změna se projeví okamžitě. Hodnota smí být 0, nebo 1.

6.11.3 Rozhraní TOKEN

Toto rozhraní umožňuje spravovat token. Obsahuje:

Pocet_vlastnosti vrátí počet vlastnosti

Vlastnost(index) – vrátí řetězcovou hodnotu této vlastnosti

Vlastnost_id(index) – vrátí index hodnoty výčtové vlastnosti. U řetězcových vlastností vrátí hodnotu -1.

Zmen_vlastnost(index, řetězec) - nastaví novou hodnotu vlastnosti. U výčtových vlastností se pokusí podle řetězce najít odpovídající hodnotu.

Zmen_vlastnost_id(index_vlastnosti, index_hodnoty) - nastaví novou hodnotu zadané vlastnosti. Nelze takto měnit hodnota řetězcových vlastností.

TRIDA – vrací číslo třídy daného tokenu.

6.11.4 Prehled

Zajišťuje získání jmen a indexů hodnot, vlastností i vzorů. Tyto indexy se pak použijí v dalších metodách rozhraní.

POCET_TOKENU - počet vzorů tokenu;

JMENO_TOKENU(index_třídy) – vrátí jméno třídy tokenu; (0<= index_třídy <počet_všech_tříd)

POCET_VLASTNOSTI(index_třídy) počet vlastnosti vzoru tokenu.

JMENO_VLASTNOSTI(index_třídy, index_vlastnosti) - jméno vlastnosti tokenu

POCET_HODNOT(index_třídy, index_vlastnosti) - počet hodnot vlastnosti tokenu

JMENO_HODNOTY(index_třídy, index_vlastnosti, index_hodnoty) - řetězec reprezentující hodnotu výčtového typu.

ID_TOKENU(řetězec) pokusí se najít číslo třídy tokenu, předanou parametrem, neúspěch => -1;parametr řetězec s hledanou třídou

ID_VLASTNOSTI(index třídy, řetězec vlastnosti) pokusí se nalézt číslo vlastnosti třídy tokenu, neúspěch => -1

ID_HODNOTY(index třídy, index vlastnosti, řetězec hodnota) - pokusí se nalézt číslo hodnoty třídy tokenu, neúspěch => -1

POCET_MIST bez parametrů, vrátí počet tříd míst.

ID_TRIDY_MISTA(název nísta) vrátí ID třídy místa, nebo -1 pokud se nenalezl vzor místa.

MÍSTO(index místa) – číslo vzoru místa, vrátí řetězec se jménem místa.

6.11.5 Rozhraní APLIKACE

Toto rozhraní umožňuje ovládat běh simulace.

Zastav – zamezí vykonávání dalších přechodů. Pozastaví simulaci.

Pridej_log(řetězec) – zápis do log konzole.

6.11.6 Rozhraní rozhrani_vstup a rozhrani_vystup

Seznam tokenů ze vstupního a výstupního rozhraní přechodu je přístupný přes tato rozhraní. Obsahují:

POCET - vrací počet tokenů v rozhraní.

TRIDA_TOKENU (index) - vrací číslo vzoru tokenu

6.12 Debugging skriptů

Aplikace sama o sobě neumožňuje debugování skriptů z důvodu využívání externího skriptovacího engine. Tento problém jde však vyřešit instalací debug prostředku pro skripty z této adresy: http://www.msdn.microsoft.com/scripting Pro umožnění debugu, který je v systému defaultně vypnut, je však zapotřebí změnit (popřípadě přidat) údaj v registru stroje: HKEY_CURRENT_USER\Software\Microsoft\Windows Script\Settings\JITDebug = 0x1

Tato změna způsobí, že kdykoliv nějaký skript v ActiveX prvku spuštěného na tomto stroji způsobí chybu, bude uživatel dotázán na spuštění debuggingu. Ve VBscriptu lze debugger vyvolat také použitím **STOP** příkazu. Pro více informací viz odkaz výše.

7 Příklad z PENTY

7.1 Výdejní automat

Asi každý se ve svém životě někdy setkal s výdejním automatem. Mohl to být například výdejní automat na kafe, na lístek na hromadnou dopravu nebo cokoliv jiného. Pro demonstraci využití Penty jsem se rozhodl, že vytvořím Petriho síť, která bude simulovat chování výdejního automatu.

7.1.1 Popis automatu

Každý výdejní automat se skládá z několika částí, jejichž funkce je u všech automatů obdobná. Mají sklad na zboží, které vydávají, mají sklad peněz, které vrací, a sklad na peníze, které přijaly. Dále u každého automatu najdeme rozhodovací logiku, která řídí chování automatu a rozhoduje, jestli může vydat zboží nebo kolik peněz má vrátit.

7.1.2 Chování automatu

K automatu náhodně přistupují zákazníci. Každý zákazník má částku peněz, kterou může utratit. Vybere zboží a vhodí dostatečný obnos peněz.

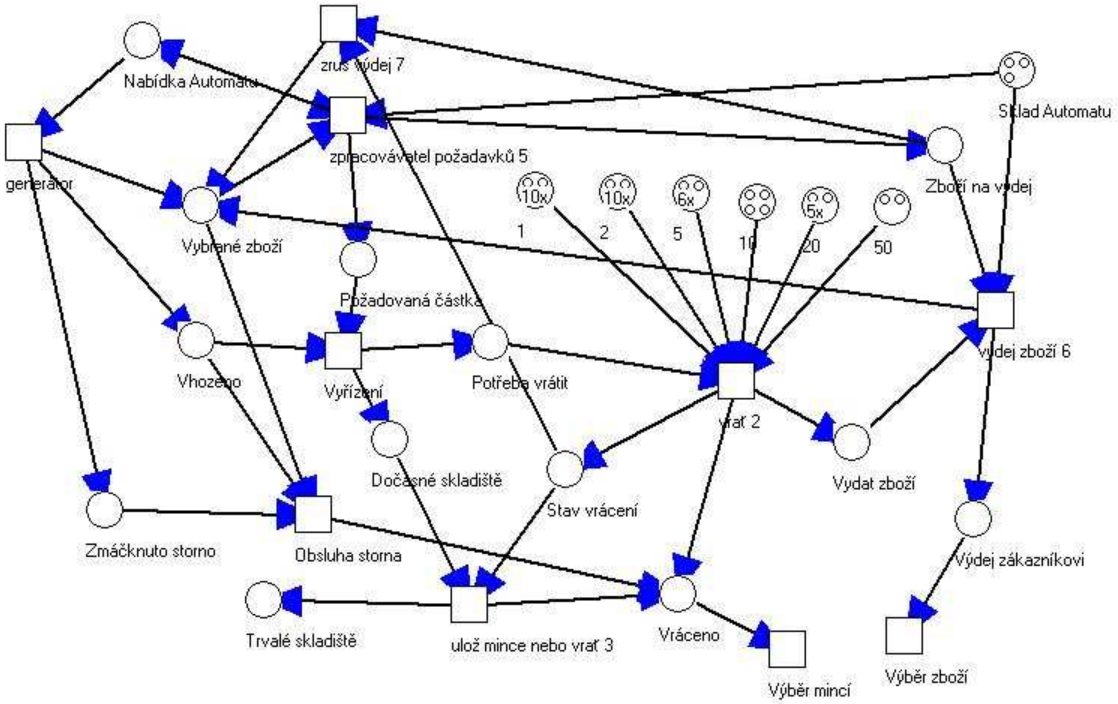
Pokud je navolené zboží a vhozen dostatečný obnos peněz, začne automat s obsluhou. Ještě než vydá zboží, zkontroluje, zda může na tuto částku vrátit. Pokud je vše v pořádku, uloží vhozený obnos do kasy, vrátí ze svých zásob přeplatek a vydá zboží. Pokud automat nemá dostatečnou zásobu peněz na vrácení, stornuje výdej a vrátí vhozené peníze.

Zákazník může stornovat celou objednávku dříve něž začne samotný výdej zboží. Vhozené peníze se vrátí.

Vydané zboží a vrácené peníze si zákazník vezme. V průběhu vydávání zboží automat nepřijímá další objednávky. Další zákazník může přijít na řadu, až když je současný zákazník obsloužen.

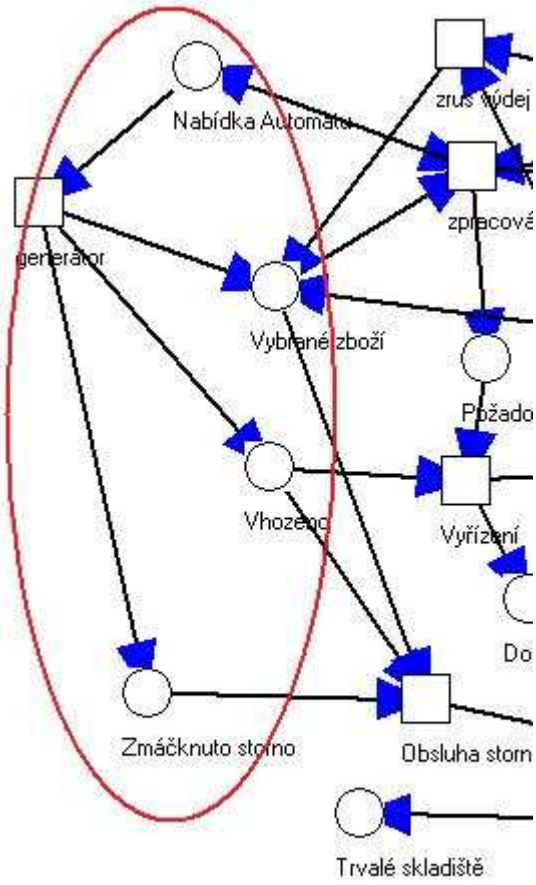
7.1.3 Navržená Petriho síť a popis implementace příkladu

Otevřete příklad „vydejni_automat.xml“.



Takto vypadá celá síť. Na první pohled může být matoucí, proto jsem níže připravil výřezy této sítě s popisem, co se v jaké části sítě odehrává.

7.1.4 Zákazník a jeho ovládání automatu



Levá strana sítě znázorňuje ovládání automatu zákazníkem. Přechod „generátor“ simuluje příchod zákazníka, který má náhodnou hotovost a podle ní si vybere zboží.

Nabízené zboží je uloženo v místě „Nabídka automatu“. Tato nabídka je pravidelně aktualizovaná podle stavu zásob automatu.

Generátor (který zastupuje zákazníka) okopíruje tokeny vybraného zboží a přesune je do místa „Vybrané zboží“. Zároveň za toto zboží zaplatí vhozením tokenů „mince“ do místa „Vhozeno“. Pokud si v průběhu volby zboží či jeho placení celou transakci

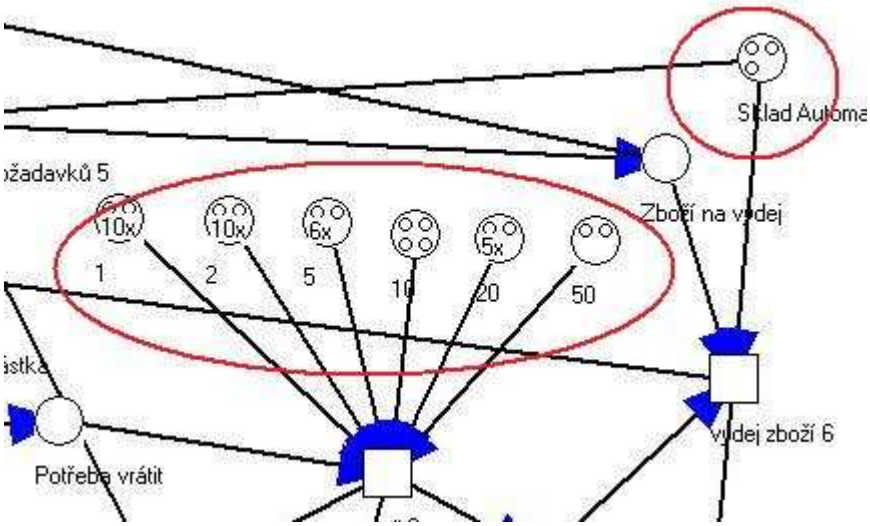
„rozmyslí“, umístí do místa „Zmáčknuto storno“ storno token. Následuje skript přechodu generátor:

```
Sub proc_generuj
  if vstupni.pocet < 1  then 'nabidka
    exit sub
  end if
  set misto_nabidka = vstupni.misto(0)

  if vystupni.pocet < 3 then exit sub 'vybrane zbozi, vhozeno a storno
  set misto_vyber = vystupni.misto(1)
  set misto_vhozeno = vystupni.misto(0)
  set misto_storno = vystupni.misto(2)

  if misto_vyber.ziskej_flag(1) =1 then exit sub 'probiha zpracovani
  if misto_nabidka.pocet =0 then exit sub 'zadna nabidka
  if misto_vyber.pocet >0 then exit sub 'nabidka je zadana...

  if misto_vyber.ziskej_flag(0) =0 and misto_vyber.pocet =0 then
    'sance, ze prijde zakaznik
     if rnd() >0.6 then exit sub
     'zacni s vyberem
     aplikace.pridej_log "zacinam vyber "
     misto_vyber.nastav_flag 0,1
     exit sub
  end if

'kolik ma zakaznik k dispozici minci:
  kc50 = Int(rnd() * 2)
  kc20 = Int(rnd() * 3)
  kc10 = Int(rnd() * 3)
  kc5 = int(rnd() * 4)
  kc2 = int(rnd() * 4)
  kc1 = int(rnd() *10)
  suma = 50 * kc50 + 20* kc20 + 10*kc10 + 5*kc5 + 2*kc2 + kc1
  suma2 = suma
aplikace.pridej_log "k dispozici je " & suma & "(" & kc50 & ", " & kc20 & ", " & kc10 & ", " & kc5 & ",
" & kc2 & ", " & kc1 & ")"

  for i=0 to misto_nabidka.pocet-1
    set token = misto_nabidka.token(i)
```

```
    cena= CInt(token.vlastnost(0))
    if (suma - cena) >=0 then
      set novy = misto_vyber.pridej(token.trida)
      novy.zmen_vlastnost 0, token.vlastnost(0)
      novy.zmen_vlastnost_id 1, token.vlastnost_id(1)
      suma=suma-cena
    end if
  next

  'probehl vyber, nyni vhodnim mince
  trida_token_mince=prehled.id_tokenu("Mince")
  cena = suma2 - suma
  aplikace.pridej_log "proběhl výběr zboží za " & cena
  'radeji preplatim, nez abych nemel mensi'
  do while (kc50> 0 and cena >= 0)
    cena = cena -50
    kc50 = kc50 -1
aplikace.pridej_log "vhazuju 50"
    set mince =misto_vhozeno.pridej(trida_token_mince)
    mince.zmen_vlastnost_id 0,5
  loop

  do while (kc20> 0 and cena > 0)
    cena = cena -20
    kc20 = kc20 -1
aplikace.pridej_log "vhazuju 20"
    set mince =misto_vhozeno.pridej(trida_token_mince)
    mince.zmen_vlastnost_id 0,4
  loop
  do while (kc10> 0 and cena > 0)
    cena = cena -10
    kc10 = kc10 -1
aplikace.pridej_log "vhazuju 10"
    set mince =misto_vhozeno.pridej(trida_token_mince)
    mince.zmen_vlastnost_id 0,3
  loop
  do while (kc5> 0 and cena > 0)
    cena = cena -5
    kc5 = kc5 -1
aplikace.pridej_log "vhazuju 5"
```

```
    set mince =misto_vhozeno.pridej(trida_token_mince)
    mince.zmen_vlastnost_id 0,2
  loop
  do while (kc2> 0 and cena > 0)
    cena = cena -2
    kc2 = kc2 -1
aplikace.pridej_log "vhazuju 2"
    set mince =misto_vhozeno.pridej(trida_token_mince)
    mince.zmen_vlastnost_id 0,1
  loop
  do while (kc1> 0 and cena >= 1)
    cena = cena -1
    kc1 = kc1 -1
aplikace.pridej_log "vhazuju 1"
    set mince =misto_vhozeno.pridej(trida_token_mince)
    mince.zmen_vlastnost_id 0,0
  loop

  misto_vyber.nastav_flag 0,0
  aplikace.pridej_log "ukoncuji vyber"
end sub
```

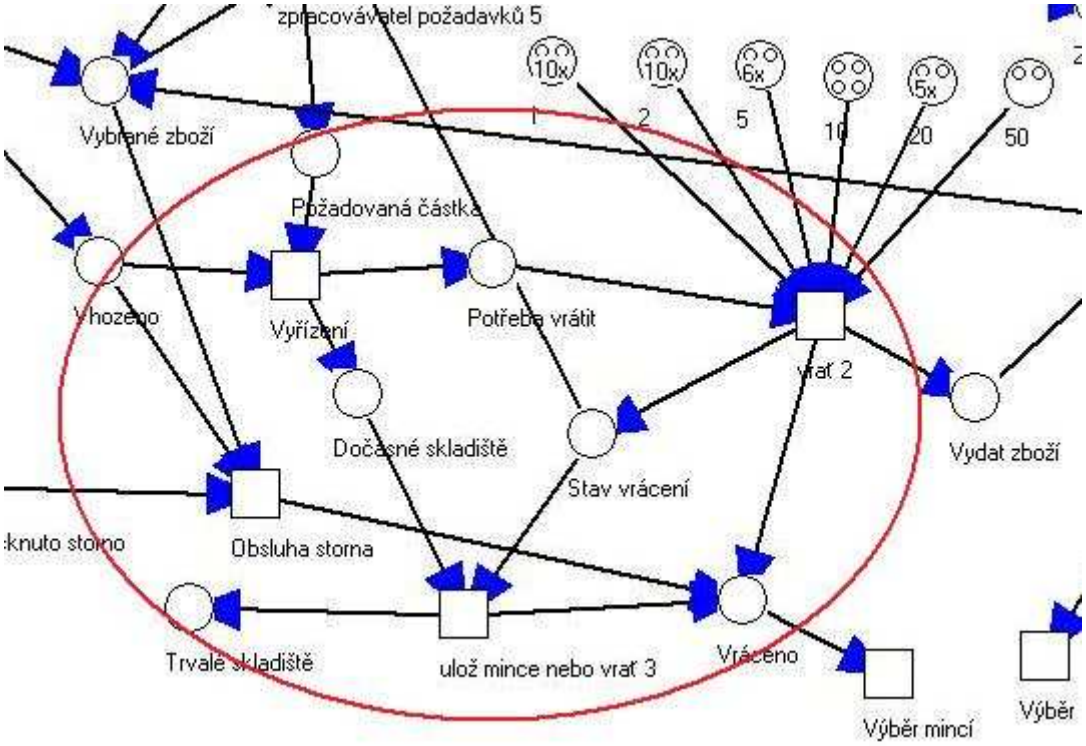
7.1.5 Sklady zboží a mincí na vrácení



Na tomto výřezu jsou sklady mincí automatu a sklad zboží. Do těchto skladů se v průběhu simulace tokeny nepřidávají, pouze ubírají. Pokud mince na vrácení nebo

zboží dojdou, žádný další zákazník nebude moct být obsloužen. Místa označené „1“, „2“, … skladují mince v dané hodnotě.

7.1.6 Rozhodovací logika automatu



Tento výřez znázorňuje logiku výdeje automatu. Z místa „Požadovaná částka“ přečte token s celkovou požadovanou částkou. Podle této sumy kontroluje sumu vhozených mincí. Pokud je dostatečná, odebere token s požadovanou částkou, přesune mince do místa „Dočasné skladiště“ a určí částku, která je potřeba vrátit.

Přechod „vrať 2“ rozhodne podle částky k vrácení a stavu mincí pro vrácení, zda má dostatek peněz na vrácení. Tento přechod přidá a nastaví úspěch (resp. neúspěch) storno tokenu, který přidá místu „Stav vrácení“. Pokud má dostatek mincí na vrácení, mince vrátí (přidá je místu „Vráceno“). Následuje skript přechodu „vrať“

```
Sub vrat
 if vstupni.pocet <7 or vystupni.pocet <3 then exit sub
 'oznaceni mist pro rychlejsi operace a prehlednost'
 set misto_vratit=vstupni.misto(6)
 if misto_vratit.pocet=0 then exit sub
 set misto1=vstupni.misto(0)
 set misto2=vstupni.misto(1)
 set misto5=vstupni.misto(2)
```

```
 set misto10=vstupni.misto(3)
 set misto20=vstupni.misto(4)
 set misto50=vstupni.misto(5)

 set misto_rozhodnuti=vystupni.misto(2)
 set misto_vydat=vystupni.misto(1)
 set misto_vrat=vystupni.misto(0)
 'urcim jakou castku potrebuju vratit'
 castka=0
‘nactu castku k vraceni
 castka=Cint(misto_vratit.token(0).vlastnost(0))
 misto_vratit.odeber(0)
 if castka<=0 then
  'nemusim nic vracet'
  cislo_tokenu=prehled.id_tokenu("storno")
  set token=misto_rozhodnuti.pridej(cislo_tokenu)
  token.zmen_vlastnost 0, "ano"
  cislo_tokenu=prehled.id_tokenu("vydat zboží")
  misto_vydat.pridej(cislo_tokenu)
  exit sub
 else
  'musim vratit'
  'odebiram nejvetsi castky a pak mensi a mensi'
  'urcim pocet 50 korun, ktere mohu odecist'
  pocet50=misto50.pocet
  if castka-pocet50*50 <0 then
   'ne vsechny padesatky pouziju'
   pocet50= castka\ 50
   castka = castka - pocet50* 50
  end if
  'podobne odectu 20'
  pocet20=misto20.pocet
  if castka-pocet20*20 <0 then
   'ne vsechny dvacitky pouziju'
   pocet20= castka\ 20
   castka = castka - pocet20* 20
  end if
  'a tak dal...'
  pocet10=misto10.pocet
  if castka-pocet10*10 <0 then
```

```
  pocet10= castka\ 10
  castka = castka - pocet10* 10
end if
pocet5=misto5.pocet
if castka-pocet5*5 <0 then
  pocet5= castka\ 5
  castka = castka - pocet5* 5
end if
pocet2=misto2.pocet
if castka-pocet2*2 <0 then
  pocet2= castka\ 2
  castka = castka - pocet2* 2
end if
pocet1=misto1.pocet
if castka-pocet1*1 <0 then
  pocet1= castka\ 1
  castka = castka - pocet1* 1
end if

if castka >0 then
  'nemam dost minci na vraceni'
  cislo_tokenu=prehled.id_tokenu("storno")
  set token=misto_rozhodnuti.pridej(cislo_tokenu)
  token.zmen_vlastnost 0, "ne"
  exit sub
else
  'mam dost penez na raceni'
  cislo_tokenu=prehled.id_tokenu("storno")
  set token=misto_rozhodnuti.pridej(cislo_tokenu)
  token.zmen_vlastnost 0, "ano"
  'nastavim vydej zbozi'
  cislo_tokenu=prehled.id_tokenu("vydat zboží")
  set token=misto_vydat.pridej(cislo_tokenu)
  'odeberu mince ze zasobniku a vratim je'
  token_mince=prehled.id_tokenu("mince")
  for i=1 to pocet50
    misto50.odeber(0)
    set token=misto_vrat.pridej(token_mince)
    token.zmen_vlastnost 0,"50"
  next
```

```
    for i=1 to pocet20
      misto20.odeber(0)
      set token=misto_vrat.pridej(token_mince)
      token.zmen_vlastnost 0, "20"
    next
    for i=1 to pocet10
      misto10.odeber(0)
      set token=misto_vrat.pridej(token_mince)
      token.zmen_vlastnost 0, "10"
    next
    for i=1 to pocet5
      misto5.odeber(0)
      set token=misto_vrat.pridej(token_mince)
      token.zmen_vlastnost 0, "5"
    next
    for i=1 to pocet2
      misto2.odeber(0)
      set token=misto_vrat.pridej(token_mince)
      token.zmen_vlastnost 0, "2"
    next
    for i=1 to pocet1
      misto1.odeber(0)
      set token=misto_vrat.pridej(token_mince)
      token.zmen_vlastnost 0, "1"
    next

  end if 'of castka >0'
 end if
end sub
```

Přechod „Ulož mince nebo vrať 3“ podle nastaveni hodnoty storno tokenu přesune vhozené mince od zákazníka buď do trvalého skladiště, to pokud se nevyskytl žádný problém, nebo je zákazníkovi vrátí, pokud se nepodařilo na jeho vhozenou sumu vrátit.

Podle nastaveni storno tokenu u místa „Stav vrácení“ se řídí výdej zboží. Pokud je vrácení v pořádku, zboží se ze skladu odebere a vydá se zákazníkovi.

Zákazník si vrácené peníze i vydané zboží odebere. Toto chování simulují přechody „Výběr mincí“ a „Výběr zboží“, které po uplynutí jednoho kola mince (resp. zboží) odeberou. Této funkčnosti se docílí tím, že se po přidání tokenů místu, na které

jsou tyto přechody napojeny, nastaví nejprve flag. Další kolo, po tom, co je flag nastaven, se odebere jak zboží (resp. mince), tak flag.

7.1.7 Řízení přístupu zákazníků

Aby se neobsluhovalo více zákazníků naráz, musí se řídit jejich přístup k automatu. Toto chování se zajišťuje nastavením flagů „Probíhá výběr“ nebo „Probíhá obsluha“ u místa „Vybrané zboží“. Nový zákazník se nevygeneruje pokud je některý z těchto dvou flagů nastaven.

Když se vygeneruje nový zákazník, nastaví se flag „Probíhá výběr“. Další kolo se vygenerují peníze a provede se výběr zboží a jeho zaplacení. Nakonec se flag „Probíhá výběr“ odebere.

Pokud místo „Vybrané zboží“ obsahuje objednávku a nemá nastaven žádný flag, značí to příchod nového zákazníka, který si vybral zboží, ale ještě nebyl obsloužen. Přechod „Zpracovatel požadavků 5“ tuto situaci rozezná, nastaví flag „Probíhá obsluha“ a spočte celkovou požadovanou sumu za vybrané zboží. Tento přechod zároveň udržuje aktuální nabídku zboží, která je zobrazena v místě „Nabídka automatu“.

8 Programové řešení

Nyní se pokusím přiblížit základní principy fungovaní aplikace. Následně popíši důležité datové struktury a chování programu zajišťující základní funkčnost PENTY.

8.1 Požadavky pro vývojové prostředí

Aplikace je napsána v Borland Delphi 7. Pro její otevření je nutné importovat do vývojového prostředí dvě Type library – jednu pro práci s XML a druhou pro komunikaci se skriptovacím rozhraním Windows Script Host. Postup jejich importu následuje.

8.1.1 Postup při importu Type library

Spusťte Delphi, aniž byste zvolili otevření nového projektu. V menu zvolte: *Project->Import Type Library*. V zobrazeném dialogu vyberte **Microsoft XML 2.0** a poté vytvořte (tlačítko Create Unit). Tento postup by měl vygenerovat novou Unitu s kódem (v tomto případě kód parseru XML dat).

Vygenerovaný obsah neměňte a zavolejte: *Components->Install New Component*. V zobrazeném dialogovém oknu zvolte druhou záložku (Into new Package). Jako název Unity zvolte cestu k Delphi\Imports\MSXML_TLB.pas (unita, která vám vznikla v předchozím kroku) Jako název balíčku pak zvolte msxml.dpk a v cestě zadejte adresář s Delphi\Lib. Zobrazí se okno, ve kterém klikněte na Install. V záložce activeX (odkud se umísťují komponenty na formuláře, při návrhu programu) by se měla ukázat nová komponenta.

Obdobně nainstalujte **Microsoft Script Control 1.0**. Pouze jména vygenerovaných unit a knihoven budou pochopitelně odlišná. Po těchto úkonech by se měl dát projekt načíst bez chyb.

8.2 Datové struktury

8.2.1 Základní třídy a proměnné v aplikaci

Aplikace je členěna do tzv. unitů. Ke každému formuláři existuje unita a kromě toho jsou v projektu další pomocné unity. Hlavní datové struktury jsou definovány v unitě *Definice*.

8.2.2 Seznamy prvků Petriho sítě

V aplikaci se setkáváme s dvěma označeními: vzory a samotné elementy Petriho sítě. Vzory se používají pro návrh elementů i jejich pozdější modifikaci. Elementy uchovávají jen ty informace, které jsou pro ně jedinečné – poloha, sousedé,… Společné informace uchovávají vzory - pojmenování, názvy hodnot, vlastností, atd.

V aplikaci se uchovávají dva typy seznamů – vzory a elementy. Seznam vzorů obsahuje řetězcové hodnoty a ukazatel na jednoho zástupce elementu. V tomto seznamu existuje právě jeden záznam pro jeden vzor. Seznamy elementů jsou pouze seznamy ukazatelů.

Seznamy vzorů se jmenují *pole_tokenu*, *pole_mist*, *pole_prechodu* a *pole_skriptu*. Všechny jsou definovány i deklarovány v unitě *Definice*.

Seznamy elementů míst a přechodů jsou uchovávány jako součást hlavního formuláře editor (unita *editor_unit*). Tyto seznamy jsou implementovány pomocí třídy TVektor, což je kontejner definovaný v unitě *Definice*.

8.2.3 Pomocné třídy

Třída TVector slouží pro uchovávání seznamu ukazatelů na třídu TPredek, což je předek pro všechny třídy elementů Petriho sítě. Tato třída se nachází v unitě Definice.

Třída je navržena tak, aby umožňovala rychlé přidávání i odebírání prvků, což se jistě hodí při běhu simulace.

TVector je implementován pomocí dvou polí - hlavního pole pro ukazatele a pomocného pole pro indexy volných míst v hlavním poli.

Myšlenka přidávání nových ukazatelů:

Pokud je pomocné pole indexu neprázdné – vezměte index na první pozici a na tento index v hlavním poli zapište nový ukazatel. Použitý index v pomocném poli nahraďte indexem z konce tohoto pole. Zmenšete počet volných indexů a zvětšete počet uložených ukazatelů. Pokud je pomocné pole prázdné, je hlavní pole bez děr. Zapište nový ukazatel za poslední ukazatel v poli.

Metoda přidání vrací index přidaného prvku v hlavním poli. Tato metoda pracuje v konstantním čase kromě případu, kdy potřebuje realokovat hlavní pole.

Odebrání prvku:

Odebírat lze dvěma způsoby – podle indexu, nebo podle ukazatele. V programu se ve většině případů odebírá pomocí indexu.

Pokud se odebírá podle indexu, ověří se, jestli je místo v hlavním poli skutečně obsazené (podle hodnoty ukazatele – volná pozice má hodnotu nil). Pokud je obsazené, vynuluje ji a přiřadí nakonec pole s volnými indexy. Odebírá se tedy v konstantním čase, až na případ, kdy je nutné zvětšit pole pro volné indexy.

Při odebírání podle hodnoty samotného ukazatele je nutné projít celé pole v lineárním čase. Pokud je ukazatel nalezen, přidá se jeho index do pomocného pole nakonec a hodnota ukazatele v hlavním poli je přepsána na nil.

Vrácení všech prvků:

Třída vrací všechny uchovávané ukazatele do pole. Existují dvě varianty – beze změny v indexaci a se změnou. Většinou je potřeba pouze výčet prvků, kdy není potřeba zachovat indexy v hlavním poli a tak i díry v něm.

Bez změny v indexaci se hlavní pole překopíruje. Se změnou v indexaci se kopírují jen platné indexy a tak se pole sesype.

8.2.4 Třídy pro elementy Petriho sítě

Všechny třídy elementů mají stejného předka: *Tpredek*. Předek má tyto nejdůležitější vlastnosti:

- id_tridy - slouží pro rozlišování různých vzorů od sebe například dvou vzorů tokenů.
- UID – číslo rozlišující dva elementy téže třídy.

Tpredek má virtuální destruktor a tyto metody:

- kopie, která vytvoří kopii třídy a vrátí ukazatel na tuto novou instanci. Pomocí kopie se tak například vytváří nové elementy.

- Save – uloží element do souboru, zápis v XML
- Load – načte element ze souboru

8.2.5 Nejdůležitější vlastnosti tříd pro elementy PS

Všechny třídy pro elementy Petriho sítě jsou přímým potomkem třídy Tpredek. Definují vlastní podoby všech jeho virtuálních metod.

Token obsahuje pole integerů - vlastnost. Každá hodnota v tomto poli vyjadřuje hodnotu jedné vlastnosti. Pole má velikost odpovídající počtu vlastností vzoru tohoto tokenu.

Místo obsahuje Tvector tokenů, omezení na tokeny (Tvyjimky), seznam vstupních a výstupních sousedů (TVector) a polohu.

Přechod je vyjádřen dvěma třídami. Tzas_prechod a Tobecny_prechod. Tobecny_prechod sdružuje společné informace pro všechny přechody stejného vzoru (totožná hodnota id_tridy) – skript, omezení na vstupní i výstupní sousedy a dvoje rozhraní tokenů. Tobecny_prechod má instance pouze v seznamu vzorů pro přechody, jinde se nevyskytuje. Tzas_prechod představuje konkrétní jeden element v PS a obsahuje jen nezbytně nutné informace: své sousedy a ukazatel na Tobecny_prechod a polohu na editační ploše.

8.3 Editační akce

V Editoru je definovaná property *akce*, která slouží pro koordinaci a ovládání editoru. Definice jejího výčtového typu je v unitě *Definice*. Hodnoty uživatel nastavuje formulářem Ovládací prvky. Pokud se přiřazuje hodnota do property akce, volá se metoda, která zároveň mění titulek editoru.

8.3.1 Postup přidávání nových prvků PS

Když se vytváří nový element PS, vytvoří se podle vzoru dané třídy kopie. (metodou kopie) Pokud je tento nový element místo nebo přechod, zařadí se do seznamu elementů editoru *mista* (resp. *prechody*) metodou *pridej*. Tato metoda vrací UID (unikátní id). Podle UID se pak dá zpětně tento element dohledat.

Editor obsahuje seznam prvků editor_prvek (unita editor_element), což je potomek třídy TImage. Instance se vykreslují na editační ploše. Editor_prvek si uchovává UID elementu, který zastupuje, i rozlišení, zda–li jde o místo, nebo přechod. Tyto zobrazovací prvky jsou uchovávány v poli *Editor.obraz*.

8.3.2 Další akce

Mazání, propojování i přidávání tokenů obstarává unita editor_element, konkrétně třída TEditor_prvek. V obslužné metodě onclick se podle hodnoty editor.akce provede příslušná činnost. Metody pro zobrazení detailu o prvku jsou v unitě Vlastnosti.

8.3.3 Primitivní grafika

O vykreslování šipek, znázorňující spojení mezi místy a přechody se stará unita system_grafika. Tato unita zodpovídá za vykreslení veškeré grafiky na ploše editoru. Kromě udržování přehledu, které šipky a zvýraznění se mají kreslit, obsahuje další pomocné metody pro správu. Označení je červené kolečko kolem aktuálně vybraného místa/přechodu.

8.4 Simulační režim

8.4.1 Spuštění simulačního režimu

Při spuštění simulačního režimu se provádí tyto akce:

- Setřídí se seznam přechodů podle priorit

- Zazálohují se všechny tokeny, pro obnovu předsimulačního stavu Petriho sítě po skončení simulace
- Zablokují se editační funkce až na přidávání a mazání tokenů
- Vymaže se veškerý kód ze ScriptControl (interface pro práci s Windows Script Host). Přidají se instance tříd pro interface aplikace a skriptů. Nakonec se nahrají procedury veškerých přechodů a pomocných skriptů.
  TODO: ukazka kodu pro ovládání skript enginu

8.4.2 Běh simulace

Simuluje se sériově. Volají se procedury přechodů od nejvyšší priority po nejmenší. Během jednoho kola se provede skript všech přechodů. Před provedením skriptu přechodu se vyhodnotí breakpoint.

Nově přiřazené tokeny jsou uzamknuty proti modifikaci. Tím pádem jsou viditelné i přístupné až v novém kole. Na konci každého kola, se všechny tokeny odemknou.

8.4.3 Ukončení simulace

Při ukončení simulace se obnoví rozložení tokenů a nastavení flagů, aby se obnovil stav před spuštěním simulace. Znovu se zpřístupní editační prvky.

8.4.4 Interface pro skripty

Aplikace využívá služeb Windows Script Host. Tyto služby v sobě sdružuje ActiveX prvek ScriptControl, který ovládá skriptovací engine a předává mu veškeré skripty, které se mají vykonat. Třídy určené pro přiřazení do ScriptControl musí implementovat rozhraní IDispatch. Tyto třídy umožňují přístup uživatelem napsaného skriptu k prostředkům aplikace – přidávat tokeny, nastavovat flagy a podobně. Všechny tyto třídy obsahuje unita Skript_interface.

Rozhraní IDispatch obsahuje čtyři funkce (GetTypeInfoCount, GetTypeInfo, GetIDsOfNames, Invoke), přičemž pro práci rozhraní používám jen poslední dvě. Z tohoto důvodu jsem vytvořil předka, který implementuje první dvě funkce rozhraní. Jeho potomci už implementují jen ty metody, ve kterých se liší.

Funkce GetIDsOfNames obdrží jako parametr řetězec a rozhodne, jestli je tento řetězec názvem nějaké metody nebo vlastnosti, ke které tato třída umožňuje přístup ze skriptu. Pokud je tento řetězec skutečně nějaká vlastnost nebo metoda, vrátí

se její kód přes parametr DISPID. Samotná funkce vrací konstantu pro zdar/nezdar. DISPID je jednoznačné číselné rozlišení metod a vlastnosti, díky němuž už není potřeba později volat metodu GetIDsOfNames.

Funkce Invoke má jako jeden z parametrů právě DISPID, který se získal voláním předchozí funkce (GetIDsOfNames). Z dalších parametrů této funkce nás zajímají tyto dva: *Params* a *VarResult.* Params je ukazatel na parametry předané společně s voláním metody. Přes *VarResult* se vrací výsledek, který může ukazovat na další instanci třídy implementující rozhraní IDispatch, a tak zajistit vícenásobné vnoření (např.: *vlastnost.podvlastnost.metoda*).

9 Příloha

9.1 Konfigurační soubor

Penta používá pro inicializaci konfigurační soubor. Tento soubor je umístěn v adresáři společně se spouštěcím souborem aplikace. Jmenuje se „config.ini“.

9.1.1 Obsah konfiguračního souboru

Konfigurační soubor se skládá z dvojic „jméno vlastnosti: hodnota vlastnosti“. Každá dvojice musí být uvedena na jednom řádku. Členy dvojce odděluje „: “ (dvojtečka a mezera). Jako poslední vlastnost je uvedeno tělo defaultní procedury. U této vlastnosti neplatí omezení pro jeden řádek a jako tělo procedury se pokládá zbytek obsahu souboru.

9.1.2 Vlastnosti nastavitelné v konfiguračním souboru

Níže je seznam vlastností a jejich přednastavených hodnot (kurzívou) následované krátkým popisem.

- *language: VBscript* – jazyk, který se má použit v rámci Windows Script Host. Zde by mohou být použity pouze nainstalované skriptovací jazyky.
- *procedure start: Sub jmeno* – hlavička procedury, která musí obsahovat „jmeno“, které se nahradí při automatickém generování skriptů přechodů
- *procedure end: end sub* – ukončení procedury
- *startX: 150* – X-ová souřadnice okna aplikace při startu (levá strana)
- *startY: 150* - Y-ová souřadnice okna aplikace při startu (horní strana)
- *width: 900* – šířka aplikace při startu
- *height: 650* – výška aplikace při startu
- *Default procedure name: Def_proc* – jméno defaultní procedury
- *DEFAULT PROCEDURE BODY:* - tělo defaultní procedury (zde neuvedeno)

9.2 Soubor projektu

Popíši XML strukturu souboru s uloženým projektem. Předem se omlouvám nenapíši zde žádný DOM, ale popíši zde strukturu více neformálně, aby si laik, který XML moc neovládá pochopil tuto strukturu.

Kořenový uzel je „Petrinet“. Tento kořenový element má č synovské uzly: „vzory“, „elementy“, „scripts“ a „scripts2“.

Vzory

Uzel „vzory“ obsahuje navržené vzory Petriho sítě. Obsahuje tyto uzly: „tokeny“, „přechody“, „místa“ a „flagy“. Uvedené uzly obsahují jednotlivé definice konkrétních vzorů.

Uzly „token“ obsahují vlastnost „nazev“ a může obsahovat uzel „vlastnosti“. Uzel „vlastnosti“ definuje vlastnosti vzoru tokenu a může obsahovat výčet hodnot.

```
<token>
    <nazev>storno</nazev>
   <vlastnost>
      <nazev>v poradku</nazev>
      <hodnota>ano</hodnota>
      <hodnota>ne</hodnota>
   </vlastnost>
 </token>
```

Uzel „prechody“ obsahuje uzly „prechod“. Každý takový uzel vyjadřuje vzor jednoho přechodu. Obsahuje název, prioritu, omezení na napojení míst, rozhraní pro tokeny, volání skriptu a vlastní skript.

```
<prechod>
    <nazev>Zalévání</nazev>
  - <vstupni_mista>
      <povoleno_vse>ano</povoleno_vse>
    </vstupni_mista>
  - <vystupni_mista>
      <povoleno_vse>ano</povoleno_vse>
    </vystupni_mista>
  - <vstupni_tokeny>
      <povoleno_vse>ne</povoleno_vse>
    </vstupni_tokeny>
  - <vystupni_tokeny>
      <povoleno_vse>ne</povoleno_vse>
      <vyjimka id="0">Zaliti</vyjimka>
    </vystupni_tokeny>
    <script>sub zalevani 'neoveruje mstupni/vystupni mista If
       vystupni.pocet=0 then exit sub if
       vystupni.misto(0).ziskej_flag(0) then exit sub 'uz sklizeno
       vystupni.misto(0).pridej(rozhrani_vystup.trida_tokenu(0))
       ''pridam zaliti end sub</script>
```

```
    <volani_scriptu>zalevani</volani_scriptu>
    <priorita>0</priorita>
  </prechod>
```

Uzel „mista“ obsahuje uzly „misto“. Obsahuje název a popis omezení na uchovávané tokeny.

```
  <misto>
    <nazev>Sklad zboží</nazev>
    <povoleno_vse>ne</povoleno_vse>
    <vyjimka id="4">zboží</vyjimka>
  </misto>
```

Uzel „Flagy“ obsahuje výčet flagů.

```
<flagy>
  <flag>Probíhá vybírání</flag>
  <flag>Probíhá obsluha</flag>
</flagy>
```

Elementy

Tento uzel obsahuje popis podoby navržené Petriho sítě.

Obsahuje dva druhy uzlů: „misto“ a „prechod“.

Uzel „misto“ obsahuje informace o své poloze, seznam flagů a tokenu s jejich ohodnocením.

```
<misto>
  <id_vzoru vzor="sklad minci">0</id_vzoru>
  <UID>3</UID>
  <X>751</X>
  <Y>169</Y>
  <pojmenovani>10</pojmenovani>
- <token>
    <id_vzoru vzor="Mince">0</id_vzoru>
    <hodnota retezec="10">3</hodnota>
  </token>
- <token>
    <id_vzoru vzor="Mince">0</id_vzoru>
    <hodnota retezec="10">3</hodnota>
  </token>
- <token>
    <id_vzoru vzor="Mince">0</id_vzoru>
    <hodnota retezec="10">3</hodnota>
  </token>
- <token>
    <id_vzoru vzor="Mince">0</id_vzoru>
    <hodnota retezec="10">3</hodnota>
  </token>
</misto>
```

Uzel „prechod“ obsahuje pojmenování, pozici, id vzoru, seznam napojených míst, úpravu priority a nastavení breakpointů.

```
<prechod>
  <id_vzoru vzor="vrať">2</id_vzoru>
  <X>737</X>
```

```
    <Y>288</Y>
    <pojmenovani />
    <vstupni>0</vstupni>
    <vstupni>1</vstupni>
    <vystupni>14</vystupni>
    <breakpoint>ne</breakpoint>
    <breakpoint_vyraz />
    <uprava_priority>0</uprava_priority>
</prechod>
```

Scripts

Obsahuje definice P-procedur.

Scripts2

Obsahuje definici procedur breakpointů