ACTA UNIVERSITATIS AGRICULTURAE ET SILVICULTURAE MENDELIANAE BRUNENSIS
SBORNÍK MENDELOVY ZEMĚDĚLSKÉ A LESNICKÉ UNIVERZITY V BRNĚ

Ročník LV 4 Číslo 3, 2007

# XML FORMÁT PRO ZÁZNAM OBJEKTOVÉ PETRIHO SÍTĚ

P. Jedlička

**Došlo: 19. prosince 2006**

**Abstract**

JEDLIČKA, P.: *XML format for notation of object-oriented Petri net*. Acta univ. agric. et silvic. Mendel. Brun., 2007, LV, No. 3, pp. 47–56

Petri nets provide executive facilities for simulation of causality, non-determinism and parallelism in discreet systems. Since they are a mathematical model in substance, they offer theory, which can be successfully used to verification of models. Executability of Petri nets predestinates them for simulation and fast prototyping. Object Petri nets represent rather complicated class, based on hierarchical and high-level Petri nets. However their complexity is balanced by their ability to identify significant characteristics of system model and to visualize it in a graphic representation.
Tools currently applied to modeling, simulation and verification of various Petri net variants use language PNML (Petri Net Markup Language) as an interchange format. However PNML is not capable of expression of object Petri net. This paper introduces prototype of XML-based language for modeling of parallel object-oriented systems described by object Petri net. This language, based on PNML, was named OPNML (Object Petri Net Markup Language).

object orientation, Petri net, XML

Historie Petriho sítí je datována od roku 1962, kdy je německý matematik C. A. Petri zavedl ve své disertační práci *Kommunikation mit Automaten* z důvodu potřeby formalismu pro popis vzájemných závislostí mezi podmínkami a událostmi modelovaného systému. Od té doby prodělala problematika Petriho sítí bouřlivý vývoj od klasických „černobílých“ P/T (Place/Transition) Petriho sítí přes C/E (Condition-Event) sítě, stochastické sítě, vysokoúrovňové (High-Level) sítě, fuzzy sítě, barvené (Coloured) sítě až po moderní objektově orientované (Object-Oriented) Petriho sítě, jež lze použít k modelování a verifikaci reálných paralelních systémů. Teorie Petriho sítí má své široké uplatnění mimo jiné v oblasti ekonomických věd. Lze je s úspěchem aplikovat při návrhu, verifikaci a simulaci různých typů ekonomických (např. logistických) systémů.

Pro popis Petriho sítě existuje přenositelný formát na bázi XML pod názvem Petri Net Markup Language (PNML) (Weber, 2006). PNML je nativním formátem uložení Petriho sítě nástroje The Petri Net Kernel (Weber, 2002), který je základem několika aplikací pro analýzu a vizualizaci Petriho sítí: inaControl, PNVis, LoLA a Petri Net Cube. Bohužel jazyk PNML poskytuje prostředky k popisu pouze nízkoúrovňových Petriho sítí. Lze jej však využít jako výchozí formát při tvorbě XML jazyka pro popis objektových Petriho sítí.

## MATERIÁL A METODY

Objektové principy jsou do Petriho sítí různými způsoby zaváděny již od počátku devadesátých let minulého století. Stručnou charakteristiku nejvýznamnějších návrhů do roku 1998 podal např. Janoušek (1998). Pro vytvoření přenositelného formátu pro popis objektových Petriho sítí (Object-Oriented Petri Nets – OOPN) bylo třeba zvolit (a částečně upravit) právě jednu z existujících koncepcí OOPN. Dále bylo nutno zmapovat situaci na poli XML jazyků, určených

buď k popisu objektově orientovaných systémů nebo Petriho sítí a zjistit, zda by nebylo vhodné vycházet při návrhu XML jazyka pro popis OOPN z některého již hotového řešení.

## Koncepce OOPN

Pro konstrukci přenositelného formátu na bázi XML pro popis objektových Petriho sítí autor vycházel z již existujících koncepcí. Byla zvolena varianta, která nejlépe splňovala následující požadavky:

- Pojetí podporuje návrh distribuovaných objektově orientovaných programových systémů tak, aby bylo možno popsanou OOPN transformovat do podoby programu v objektově orientovaném jazyce – především za účelem simulace modelovaného systému.
- Objektová Petriho síť umožňuje reprezentaci tříd a jejich instancí objektově orientovaného programového systému s možností definice datových položek, metod a událostí.
- Poskytuje možnost hierarchizace navrhovaného systému.
- Základní vlastností sítě je koncept rozlišitelnosti značek sítě, jež reprezentují objekty modelovaného systému s jejich atributy.
- Přímá podpora možnosti návrhu distribuovaných programových systémů, jejichž každá komponenta může pracovat s několika vlákny programového systému (podpora paralelismu jako jedné ze základních vlastností Petriho sítí).
- Třída objektových Petriho sítí je postavena na formální matematické teorii, jež poskytuje prostředky pro konstrukci sítě reprezentující studovaný systém a jeho analýzu.
- Pojetí OOPN je použitelné pro modelování ekonomických a logistických systémů.

Uvedené požadavky splňuje a za základ definice objektové Petriho sítě byla použita koncepce I. Martiníka (1999), dále bylo využito některých principů popsaných v (Janoušek, 1998). Vzhledem ke značnému rozsahu specifikace OOPN nelze její kompletní popis uvést v tomto příspěvku, zájemce lze ale odkázat na webovou stránku autora věnovanou jazyku OPNML: http://www.pef.mendelu.cz/~petrj/opnml. Základní struktura OOPN je následující:

Vzhledem k tomu, že byl dán požadavek převoditelnosti OOPN na objektově orientovaný programový systém, pracujeme zejména s třídami a jejich instancemi (objekty). Deklarace a definice třídy je v OOPN vyjádřena dvěma stránkami sítě:

Stránka třídy (class page) reprezentuje datové položky a metody třídy (datové položky a metody typu static) – tato stránka existuje v jediné instanci pro všechna značení objektové Petriho sítě.

Stránka instance třídy, tzn. stránka objektu (class instance page, object page), reprezentuje datové položky, metody a události instance třídy – počet instancí této stránky je totožný s počtem existujících instancí dané třídy v příslušném značení sítě. Instance stránek objektu jsou dynamicky vytvářeny a rušeny spolu s vytvářením a rušením objektů programového systému, jež reprezentují. **Datové položky** třídy nebo její instance jsou reprezentovány **značkami** Petriho sítě. Každé značce sítě jsou jednoznačně přiřazeny: identifikátor, datový typ, viditelnost (public, protected nebo private) a hodnota.

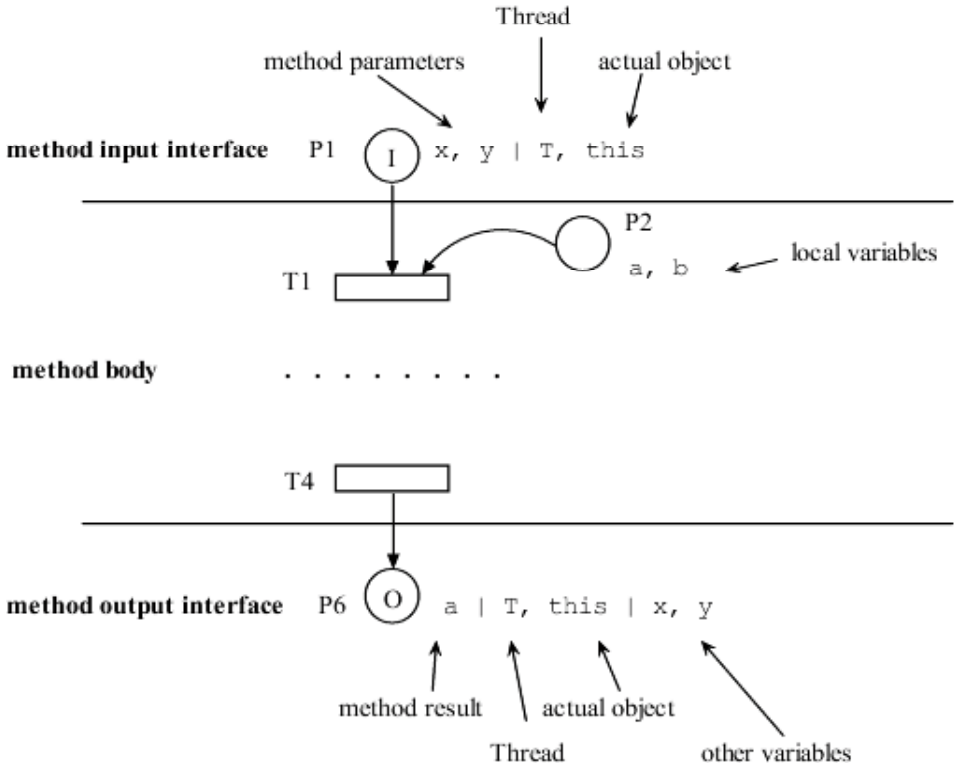
Každá z **metod** třídy nebo instance je reprezentována samostatnou podstránkou metody, jež je složená ze vstupního interface, těla podstránky a výstupního interface (viz obr. 1). Instance podstránky metody dynamicky vzniká a zaniká při volání metody, již reprezentuje. Každá stránka OOPN je pak složena z:

- hlavní podstránky reprezentující datové položky třídy nebo její instance,
- podstránek metod reprezentujících metody třídy nebo její instance.

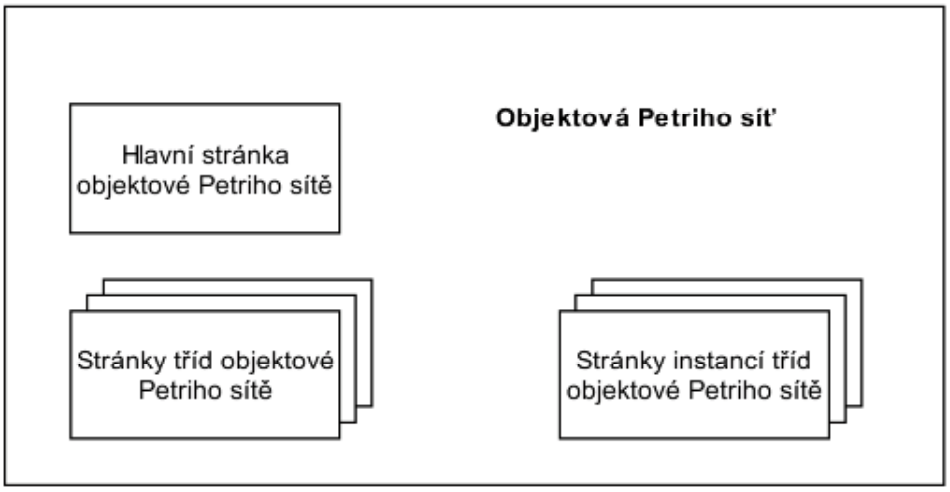
## Struktura objektové Petriho sítě

Objektová Petriho síť má jako celek strukturu zachycenou na obr. 2. Spuštění objektově orientovaného programového systému reprezentovaného touto sítí je vyjádřeno vytvořením instance hlavní stránky sítě. Prostřednictvím hierarchických přechodů hlavní stránky a poté i hierarchických přechodů instancí dalších stránek pak dochází k vytváření objektů (tzn. instancí stránek instancí tříd) a volání metod (tzn. vytváření instancí podstránek metod). Pro určitý stav modelovaného systému tak existuje i odpovídající stav objektové Petriho sítě a jejího značení. Právě toto její značení pak odráží modelované objekty, hodnoty jejich atributů a stav běžících metod programového systému.

Změna stavu OOPN, stejně jako v ostatních variantách Petriho sítí, je realizována prostřednictvím přechodů, které jsou spojeny s místy sítě prostřednictvím vstupních a výstupních hran. Významnou charakteristikou **hran** je hranová funkce, tvořená v OOPN množinou výrazů. Proměnnými jednotlivých výrazů jsou značky podstránky metody, jíž je hrana prvkem. Jedná-li se o *vstupní hranu přechodu*, jsou všechny výrazy hranové funkce typu *boolean*. Jsou-li hodnoty všech výrazů po navázání příslušných značek a jejich vyhodnocení rovny hodnotě *true* u všech vstupních hran (ne inhibičních) přechodu, smí být daný přechod proveden. Není-li tomu tak (tj. alespoň jeden výraz nabude po svém vyhodnocení hodnotu *false*), přechod nesmí být proveden pro dané vázání značek. U *výstupních hran přechodu* pak výrazy vyjadřují změnu hodnoty příslušné navázané značky (tedy změnu atributu objektu nebo třídy).




1: *Tvar rozhraní podstránky metody*




2: *Struktura objektové Petriho sítě*

**Přechod** v OOPN má na rozdíl od P/T Petriho sítí jednu významnou charakteristiku: každý přechod smí být asociován s nejvýše jednou výstupní hranou. Kromě „obyčejných“ přechodů může OOPN obsahovat i hierarchické přechody, reprezentující podstránky metod. Provedení hierarchického přechodu tedy není atomická operace, ale představuje vytvoření podstránky metody, navázání vstupních značek přechodu na vstupní interface podstránky, realizaci všech kroků této podstránky, navázání značek z výstupního interface podstránky na výstupní značky přechodu a následné zrušení podstránky metody. Rozlišujeme *synchronní* a *asynchronní hierarchický přechod*. U synchronního přechodu dochází k umístění jeho výstupních značek do výstupního místa až po zrušení podstránky metody (tzn. po dokončení její činnosti), zatímco asynchronní přechod na dokončení kroků podstránky metody nečeká a ukládá výstupní značky do výstupního místa okamžitě po vytvoření podstránky metody.

## XML jako formát pro popis objektových Petriho sítí

Jazyk XML, který vznikl odvozením od jazyka SGML, představuje množinu pravidel pro tvorbu značkovacích jazyků použitelných k popisu v podstatě libovolných struktur. Pro své dobré vlastnosti, mezi které patří především relativní jednoduchost, přenositelnost a snadná manipulace daná textovým formátem, si získal značnou oblibu, o čemž svědčí řada jeho aplikací (W3C, 2006). Formát XML byl tedy zvolen i pro popis objektových Petriho sítí.

Jako výchozí formát byl použit jazyk **PNML**, vytvořený v Institutu informatiky Humboldtovy univerzity v Berlíně pod vedením Michaela Webera (Weber, 2006). Jazyk PNML je nativním formátem uložení Petriho sítě v nástroji The Petri Net Kernel. Formát PNML je rovněž podporován nástrojem CPN Tools[1], vytvořeným výzkumnou skupinou CPN group v Department of Computer Science, Faculty of Science, University of Aarhus v Dánsku. Pro definici XML jazyka schopného popsat Petriho síť tvůrci použili schémový jazyk Relax NG. Byla vytvořena definice základního jazyka pro nehierarchickou Petriho síť s názvem *basic PNML* a následně její další dvě rozšíření pro strukturovanou (*structured PNML*) a modulární Petriho síť (*modular PNML*).

Dalšími formáty, kterými bylo možné se částečně inspirovat, jsou jazyky XMI a SOX. Metajazyk **XMI** (XML Metadata Interchange) byl vytvořen pro účely nezávislého a otevřeného vyjádření modelů popsaných vizuálním jazykem UML. V současné době je základním de facto standardem výměny UML modelů mezi modelovacími nástroji a současně jedním z prostředků jejich perzistentní reprezentace. Jazyk XMI je navržen a spravován skupinou OMG (Object Management Group)[2].

Jazyk **SOX** (Schema for Object-Oriented XML) (W3C, 2004) je schémovým jazykem podobně jako WXS nebo Relax NG. Jeho specialitou je využití objektových principů dědičnosti a polymorfismu při definici XML jazyka. Ta má (na rozdíl od DTD a shodně např. s WXS) podobu XML dokumentu. Dědičnost je realizována pomocí elementu *extends*, jehož atribut *type* obsahuje jméno elementu, jehož struktura se dědí.

Dříve, než bylo možné začít vytvářet vlastní XML jazyk pro popis objektových Petriho sítí, bylo nutné nejdříve zvolit formát jeho definice. To znamenalo vybrat některý z existujících schémových jazyků a seznámit se s jeho syntaxí a sémantikou. Obvyklými kritérii výběru jsou šíře podpory u XML nástrojů, jednoduchost a výstižnost jazyka a také jeho modelovací schopnosti s ohledem na charakter modelované entity nebo systému. V našem případě bychom mohli hledat jazyk, který je nějakým způsobem disponován pro modelování objektově orientovaných systémů a jejich typických vlastností a/nebo jazyk, který by se hodil pro popis Petriho sítí. Třemi nejrozšířenějšími obecnými schémovými jazyky pro XML jsou DTD, XML Schema (WXS) a Relax NG. Z jazyků specificky zaměřených jsme si stručně představili jazyky XMI a SOX.

Jazyk **XMI** je speciální schémový jazyk pro tvorbu DTD definujících přenositelný XML formát modelů UML. Jedná se o poměrně složitý jazyk, který je prakticky jednoúčelový. Jeho použití mimo UML je neefektivní až nemožné, pokud příslušný formalismus neumožňuje konstruovat diagramy známé z UML. Použití XMI pro popis objektových Petriho sítí by bylo vhodné pouze v případě, pokud bychom jejich principy nějakým způsobem uplatnili v UML. Takové tendence se již v uplynulých letech několikrát objevily (Hu, Shatz, 2004; Baresi, Pezze, 2001), ale žádné ucelené řešení integrace Petriho sítí do UML dosud zpracováno nebylo.

Jazyk **SOX** má sice implementovány objektové principy dědičnosti a polymorfismu, ale ty jsou uplatnitelné pouze v definici jazyka, nikoliv v jazyce samotném.

K definici XML jazyka pro objektové Petriho sítě byl tedy vybrán jeden z obecných schémových jazyků. Ostatně stejný přístup zvolili tvůrci výše popsaného jazyka PNML, když použili jazyk Relax NG.

Nejstarší alternativou v tomto ohledu je jazyk **DTD**, který je ale objektivně nejhorší možnou variantou. Při definici struktury elementu například neumožňuje přesnější určení počtu subelementů, nepodporuje také vlastní definice datových typů elementů nebo atributů. Navíc definice jazyka ve formě DTD nemá formát XML, takže ji nelze parsovat (syntaktickou analýzou ověřovat) oproti vlastní definici typu dokumentu.

Jazyk **Relax NG** má již podobu XML jazyka a také disponuje širšími modelovacími možnostmi. Navíc byl již použit k definici jazyka PNML, na kterou by bylo možné navázat. Autor tohoto příspěvku se však z praktických důvodů rozhodl upřednostnit kritérium co nejširší podpory jazyka u nástrojů pro práci s XML a zvolil jazyk **WXS** (XML Schema). Při použití obecného schémového XML jazyka navíc není problém definici XML jazyka transformovat do podoby jiného schémového jazyka. Obousměrnou konverzi mezi

1 http://wiki.daimi.au.dk/cpntools/cpntools.wiki
2 http://www.omg.org

DTD a WXS provádí např. program XMLSpy firmy Altova[3], převod mezi formáty DTD, RelaxNG a WXS lze provést pomocí programu Trang[4].

Výběrem schémového jazyka však ještě není dán způsob konstrukce nového jazyka. Postupem času se vyvinulo několik přístupů k výstavbě schématu. Problém, který je potřeba rozhodnout, spočívá ve výběru, kdy používat lokální a kdy globální elementy a jak moc má být schéma rozšiřitelné a použitelné v dalších schématech. První přístup je označován jako *matrjóška*, tedy ruská panenka, kde do sebe zapadají jednotlivé menší a menší panenky. V tomto přístupu je globální jen jeden element a všechny ostatní jsou uvnitř něj definovány jako lokální – viz následující příklad:

```
<xs:element name="zamestnanec">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="jmeno" type="xs:string" />
      <xs:element name="prijmeni" type="xs:string" />
      <xs:element name="plat" type="xs:decimal" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Druhý, rovněž často používaný přístup, se nazývá *salámová kolečka*. Všechny elementy jsou definovány jako globální a pak jsou složeny dohromady pomocí odkazů. Element, použitý ve struktuře jiného elementu, může být totiž definován samostatně a do struktury zahrnut prostřednictvím odkazu v atributu *ref*:

```
<xs:element name="jmeno" type="xs:string" />
<xs:element name="prijmeni" type="xs:string" />
<xs:element name="plat" type="xs:decimal" />

<xs:element name="zamestnanec">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="jmeno" />
      <xs:element ref="prijmeni" />
      <xs:element ref="plat" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Dokument může začínat libovolným elementem, libovolný element je možno znovu použít. Nevýhoda je, že v tomto přístupu nelze pro element stejného názvu definovat dva různé modely obsahu v závislosti na kontextu jeho výskytu. Tuto možnost předchozí způsob nabízí.

Třetí metoda, označovaná jako *metoda slepého Benátčana*, pro všechny elementy nejprve definuje typy, které lze znovu používat. Elementy jsou pak definovány lokálně pomocí těchto typů, takže mohou mít při shodě jmen různé modely obsahu. Tento přístup nabízí nejvíce možností a komfortu a ve většině případů je nejvhodnější. Jeho nevýhodou je větší pracnost a složitost v porovnání s předchozími metodami.

Protože každý z uvedených postupů má své kladné i záporné stránky, byl vždy volen optimální postup na základě konkrétní situace. Pro definici značkovacího jazyka určeného k záznamu objektových Petriho sítí byly stanoveny následující zásady:

- Element, který bude pod stejným názvem použit jako součást struktury několika dalších elementů, definujeme samostatně a na tuto definici se budeme odkazovat atributem *ref*.
- Pokud budeme potřebovat více elementů různých názvů, ale stejné struktury, definujeme tuto strukturu jako pojmenovaný komplexní datový typ a na něj se pak budeme odkazovat v atributu *type*.
- Pro různé elementy, které budou mít společnou významnou část své struktury, definujeme pojmenovaný komplexní datový typ, který posléze použijeme jako základ definice struktury každého z nich.

## VÝSLEDKY

Ve formátu WXS byla vytvořena definice jazyka na bázi XML určeného pro záznam objektové Petriho sítě. Jazyk byl podle vzoru PNML nazván Object-oriented Petri Net Markup Language (ve zkratce OPNML). Kompletní definice jazyka OPNML včetně metodiky jeho použití pro popis objektové Petriho sítě je k dispozici na webové stránce http://www.pef.mendelu.cz/~petrj/opnml. Jazyk disponuje konstrukty pro popis všech komponent a vazeb sítě. Umožňuje modelovat i objektová rozhraní, testovací hrany a hierarchické stránky, třídy a metody lze definovat i jako abstraktní nebo finální.

Modelování tradičních vlastností objektově orien-

3 http://www.altova.com/products_ide.html
4 http://www.thaiopensource.com/relaxng/trang.html

tovaných systémů podporuje OPNML následujícím způsobem:

- Dědičnost je modelována volitelným atributem *extends* elementu definice třídy *classDef*. Hodnotou atributu je jméno nadtřídy.
- K určení stupně zapouzdřenosti slouží atribut *visibility* s oborem hodnot *public*, *protecte* a *private*.
- Hierarchické přechody, které reprezentují podstránky metod a destruktorů, nemají pevně stanovenu třídu spouštěné metody, ale jen její název a seznam skutečných parametrů. Konkrétní třída, jejíž metoda daného jména bude v rámci hierarchického přechodu vytvořena, je určena až na základě skutečného typu (nikoliv bázového) referenční proměnné (tzn. značky), jehož název (tj. jméno třídy) je hodnotou atributu *actualType* elementu značky *token*. Tento princip odpovídá požadavku modelování polymorfismu jazykem OPNML.

Kromě toho, že jazyk OPNML zachycuje strukturu i sémantiku objektové Petriho sítě, je schopen popsat i její graf (polohu a vizuální parametry komponent). Parametry textu, používaného pro popis grafu, i obory jejich hodnot jsou převzaty z kaskádových stylů CSS2.

Zvláštní pozornost byla věnována popisu hranových funkcí. Vhodné elementy pro jejich popis byly nalezeny v jazyce MathML[5]. Tento jazyk mohl být využit ve stávající podobě a elementy popisující především operátory výrazů hranových funkcí mohly ležet ve jmenném prostoru jazyka MathML. Důvody, proč tak nebylo učiněno, jsou v zásadě dva:

- Definice jazyka MathML nezohledňuje povinný počet operandů jednotlivých skupin operátorů (unárních, binárních, *n*-árních). V jazyce MathML lze vytvořit dokument, který je oproti definici jazyka validní, avšak z matematického hlediska zcela nesmyslný. Elementy operátorů jsou totiž definovány bez jakéhokoliv kontextu.
- Většina potřebných elementů jazyka MathML jsou prázdné elementy, takže jejich vlastní definice příliš nezkomplikovala definici jazyka OPNML.

Hranové funkce vstupních a výstupních hran přechodů se liší syntaxí i sémantikou. *Vstupní hrany* přechodů mají přiřazeny logické výrazy, na jejichž vyhodnocení závisí proveditelnost přechodů. *Výstupní hrany* jsou popsány přiřazovacími příkazy, kterými se mění hodnoty značek sítě. Každý přiřazovací příkaz se skládá z názvu značky, operátoru přiřazení a výrazu, jehož typ je kompatibilní s typem značky. Jazyk OPNML pro tyto účely disponuje čtyřmi elementy operandů, patnácti elementy pro záznam operátorů a elementem *apply*, přejatým z jazyka MathML, jenž reprezentuje závorky uzavírající výraz. Pro reprezentaci celého výrazu hranové funkce je připraven element *expression*. Jeho nepovinný subelement *lValue* (left value, levá hodnota) slouží k určení značky, které je výsledek výrazu přiřazen. Výskyt tohoto elementu znamená, že se jedná o hranový výraz výstupní hrany přechodu. Naopak jeho absence indikuje výraz vstupní hrany.

Každý definovaný element budovaného jazyka, který má alespoň jeden atribut nebo subelement, je popsán komplexním datový typem. Jediným zástupcem *pojmenované skupiny elementů* v definici jazyka OPNML je skupina *anyElement*. Skládá se z jednoho elementu libovolného názvu i typu. Atribut *namespace* tohoto elementu svou hodnotou *##other* vymezuje element do jiného jmenného prostoru, než je jmenný prostor elementů jazyka OPNML. Elementy bez předem určeného jména a typu mohou být součástí struktury elementu *toolSpecific*. Ten slouží k uložení specifických informací softwarových nástrojů pracujících s modelem a počet jeho výskytů není omezen. Element *toolSpecific*, převzatý v nezměněné podobě z jazyka PNML, je volitelnou součástí struktury všech elementů jazyka OPNML. Jeho textové atributy *tool* a *version* slouží k identifikaci programu, který si k danému elementu specifickou informaci uložil, a jeho verze. Element *toolSpecific* je definován takto:

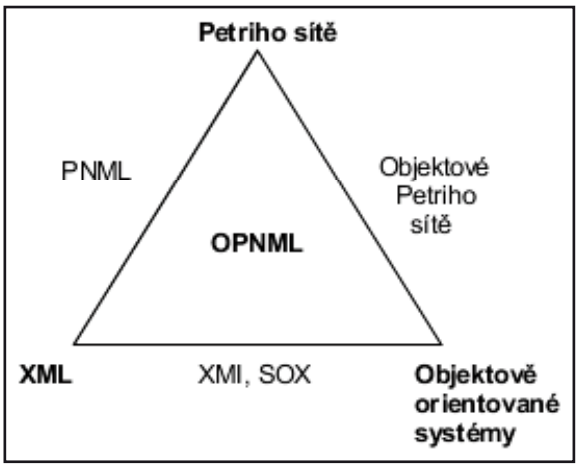
```
<xs:element name="toolSpecific">
  <xs:complexType>
    <xs:group ref="anyElement" minOccurs="0" maxOccurs="unbounded" />
    <xs:attribute name="tool" type="xs:string" use="required" />
    <xs:attribute name="version" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
```

## DISKUSE

Byl vytvořen prototyp jazyka na bázi XML pro modelování paralelních objektově orientovaných systémů popsaných objektovou Petriho sítí. Jazyk dostal název OPNML (Object Petri Net Markup Language) a vznikl na základech jazyka PNML, který je v současné době uznávaným a de facto jediným používaným

5 http://www.w3.org/TR/MathML/

výměnným formátem (neobjektových) Petriho sítí. OPNML je ucelený formát schopný popsat jak strukturu a sémantiku vysokoúrovňové hierarchické Petriho sítě, tak všechny podstatné vlastnosti objektově orientovaných systémů – dědičnost, zapouzdřenost i polymorfismus. Výsledek práce propojuje tři oblasti výzkumu: Petriho sítě, objektově orientované systémy a značkovací jazyky na bázi XML (viz obr. 3).




3: *Pozice jazyka OPNML*

Definice jazyka OPNML, jakožto XML aplikace, má sama o sobě rovněž podobu XML dokumentu se všemi výhodami této formy uložení strukturovaných dat. Jazyk OPNML je definován jako univerzální, bez podpory případného specifického zaměření modelovaného systému. Z tohoto důvodu je třeba počítat s potřebou budoucího rozšíření jeho modelovacích schopností. Toho je možné docílit v zásadě dvěma způsoby. Buď lze využít volnosti, kterou definice jazyka poskytuje pro ukládání vlastních údajů softwarovým nástrojům, nebo je možné jazyk doplnit o potřebné definice.

Bohužel se nepodařilo docílit plné kompatibility jazyka OPNML s jazykem PNML, a to především proto, že PNML neumožňuje modelování dynamických systémů a současně v objektových Petriho sítích se nelze obejít bez dynamické instanciace stránek sítě. OPNML je s PNML kompatibilní pouze na úrovni jednotlivých stránek modelu objektového systému.

### Přínosy práce pro teorii a praxi

Za hlavní přínos jazyka OPNML lze označit návrh metodiky současného zobrazení struktury a stavu objektově orientovaného programového systému, popsaného objektovou Petriho sítí, ve značkovacím jazyce typu XML. V jednom dokumentu je tak možné v nezávislém strukturovaném formátu modelovat:

a) strukturu i sémantiku objektového programového systému v podobě hierarchického systému tříd,
b) libovolný stav běžícího programu dosažitelný ze stavu počátečního (v terminologii Petriho sítí se jedná o tzv. dosažitelné značení).

Schopnost zaznamenat okamžitý stav modelovaného systému v nezávislém formátu umožňuje zkoumat tento stav i systém v různých nástrojích, většinou odlišně disponovaných. Nástroj zaměřený na simulaci může modelovat další vývoj systému, formální analyzátor oproti tomu určí vlastnosti Petriho sítě, od nichž lze odvodit vlastnosti modelovaného systému. V případě objektové Petriho sítě mohou mít její vlastnosti následující vypovídací schopnost:

- **Omezenost sítě** – má vztah k paměťovým nárokům modelovaného objektového programového systému. Umožňuje např. identifikovat nebezpečí vzniku nekonečné rekurze volání metod, která by vedla k zahlcení paměti a deadlocku.
- **Živost sítě** – lze zjistit, jestli a za jakých okolností může za běhu programu dojít k „odumření“ některé jeho části, tzn. k zablokování možnosti jeho dalšího využití. Taková část programu se v objektové Petriho sítí vyznačí jako *neživá*.
- **Živost jednotlivých přechodů sítě** – je-li jako neživý zjištěn např. hierarchický přechod reprezentující v modelu určitou metodu, nemůže za žádných podmínek dojít ke spuštění této metody. Taková metoda buď není potřeba a je nadefinována zbytečně, nebo je chyba v algoritmu programu.

Pokud je analýzou objektové Petriho sítě, provedenou určitým nástrojem, zjištěn nějaký nedostatek modelovaného systému, lze model přenést do nástroje vhodnějšího pro úpravu modelu, zde provést potřebné změny, poté upravený model přenést zpět a opětovným určením vlastností ověřit efekt úpravy.

Vlastnosti objektově orientovaných systémů byly do Petriho sítě zaneseny pouhou úpravou některých komponent (hierarchické přechody, I/O místa, …), takže se v objektové Petriho sítí nevyskytují žádné cizorodé prvky. Výhodou tohoto přístupu je, že objektová Petriho síť nevyžaduje zvláštní způsob zacházení a je tudíž v principu zpracovatelná současnými softwarovými nástroji pro práci s Petriho sítěmi.

Snahou bylo přenést model objektově orientovaného systému modelovaný objektovou Petriho sítí do sféry jazyků XML a zůstat zde pokud možno i při zpracování modelu. Výhodou platformy XML je možnost využití celé škály již existujících nástrojů a metod bez nutnosti používat speciální programové vybavení. Tato přednost byla prokázána relativní snadností transformace objektové Petriho sítě z jazyka OPNML do grafu ve formátu SVG, kdy stačilo vytvořit transformační šablonu v jazyce XSLT a samotnou transformaci již provedl XSLT procesor, tedy standardní nástroj z oblasti XML jazyků. Výhody formátu

XML se však mohou projevit již při tvorbě modelu. V definici jazyka je možné určit pravidla, která pak musí dokument v daném jazyce splňovat. Zda dokument tato pravidla neporušuje, lze zjistit standardním nástrojem nazvaným XML parser, který dokument označí buď za platný *(valid)* nebo neplatný *(invalid)*. Například jazyk OPNML vyžaduje existenci minimálně jednoho přechodu a dvou hran v každé podstránce metody. Porušení tohoto pravidla by znamenalo, že postránka sítě nesplňuje základní podmínky konstrukce grafu Petriho sítě.

### Možné další směry vývoje

Varianta objektových Petriho sítí, která tvořila bázi pro definici jazyka OPNML, pokrývá pouze základní potřeby objektově orientovaných programových systémů. Autor práce počítá s následným rozvojem jazyka tak, aby byl schopen postihnout další konstrukce a principy často používané v objektových programovacích jazycích. Jedná se například o:

- **Modelování zřetězení volání metod** – v současné době se často používá volání více metod v rámci jednoho příkazu, přičemž návratová hodnota jedné metody je objektem volajícím metodu následující. Tento princip je v aktuální podobě jazyka OPNML nutno nahradit postupnou realizací hierarchických přechodů, které si své vstupně/výstupní hodnoty předávají prostřednictvím pomocných proměnných (značek).
- **Použití metod jako skutečných parametrů jiných metod** – parametrem metody v OPNML může být pouze proměnná nebo konstanta. Běžně se však v programovacích jazycích na místě skutečných parametrů metod používají i jiné metody. I tento princip lze v OPNML nahradit pomocnou proměnnou.
- **Vznik výjimek a jejich obsluha** – vznik výjimky v běhu programu má zpravidla podobu vytvoření instance některé ze speciálních tříd (např. v jazyce Java mají většinou v názvu slovo *Exception* – výjimka). Po vzniku výjimky běh programu nepokračuje následující instrukcí, ale blokem příkazů, určených k ošetření výjimky.
- **Vznik událostí a reakce na ně** – mnoho moderních aplikací je řízeno událostmi. Jejich vznik a obsluha jsou podobné vzniku a ošetření výjimky.

Na závěr nelze alespoň nezmínit některé tendence z posledních let, které se snaží o spojení výhod objektových Petriho sítí v oblasti simulace a verifikace paralelních systémů a jazyka UML, jakožto současného standardu pro objektové modelování (Hu, Shatz, 2004; Baresi, Pezze, 2001). Přednosti obou formalismů se zdají být komplementární a jejich účelová kombinace tudíž přínosná. Jelikož jazyk UML disponuje vlastním přenositelným formátem uložení modelů – jazykem XMI, může nastat situace, která bude směřovat ke sjednocení obou formalismů i na úrovni výměnných formátů.

SOUHRN

Petriho sítě poskytují výkonné prostředky pro modelování kauzality, nedeterminismu a paralelismu v diskrétních systémech. Protože jsou ve své podstatě matematickým modelem, nabízejí teorii, která může být úspěšně využita pro verifikaci modelů. Proveditelnost Petriho sítí je předurčuje pro simulaci a rychlé prototypování. Objektové Petriho sítě představují dosti komplikovanou třídu, založenou na hierarchických a vysokoúrovňových Petriho sítích. Jejich složitost je však vyvážena jejich schopností určit významné vlastnosti modelovaného systému a zobrazit jej v grafické podobě.
V současnosti používané nástroje na modelování, simulaci a verifikaci různých variant Petriho sítí používají pro vzájemnou výměnu modelů formát jazyka PNML (Petri Net Markup Language), který však není disponován k vyjádření objektové Petriho sítě. Tento příspěvek představuje prototyp jazyka na bázi XML pro modelování paralelních objektově orientovaných systémů popsaných objektovou Petriho sítí. Jazyk, který vznikl na základech jazyka PNML, byl pojmenován OPNML (Object Petri Net Markup Language) – viz http://www.pef.mendelu.cz/~petrj/opnml.

objektová orientace, Petriho sítě, XML

Tento příspěvek vznikl v rámci řešení výzkumného záměru VZ MSM 6215648904/03/03/04.

### LITERATURA

BARESI, L., PEZZE, M.: Improving UML with Petri nets. In *Electronic Notes in Theoretical Computer Science.* Vol. 44, 2001. [online] [cit. 8. 7. 2006]. URL: <http://www1.elsevier.com/gej-ng/31/29/23/73/28/52/44.4.010.pdf>.

HU, Z., SHATZ, S. M.: Mapping UML Diagrams to

a Petri Net Notation for System Simulation. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE), Ban, Canada, June 2004*. [online] Poslední aktualizace 29. 6. 2004. [cit. 10. 12. 2006]. URL: <http://www.cs.uic.edu/ shatz/papers/seke04.pdf>.

JANOUŠEK, V.: Modelování objektů Petriho sítěmi. 1. vyd. Brno: VUT, 1998. 137 p. Disertační práce.

MARTINÍK, I.: *Metodologie tvorby objektově-orientovaných programových systémů s využitím teorie objektových Petriho sítí.* 1. vyd. Ostrava: VŠB-TU, 1999. 218 s. Disertační práce.

W3C – World Wide Web Consortium: *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. [online] Ver. 1.0, poslední aktualizace 29. 11. 2006. [cit. 13. 12. 2006]. URL <http://www.w3.org/TR/2006/REC-xml-20060816/>.

W3C – World Wide Web Consortium: *Schema for Object-Oriented XML 2.0.* [online] Ver. 2.0, poslední aktualizace 30. 7. 2004. [cit. 13. 12. 2006]. URL: <http://www.w3.org/TR/NOTE-SOX/>.

WEBER, M.: *Petri Net Kernel*. [online] Poslední aktualizace 6. 2. 2002. [cit. 13. 12. 2006]. URL <http://www.informatik.hu-berlin.de/top/pnk/index.html>.

WEBER, M.: *Petri Net Markup Language* [online] [cit. 2006-12-13]. URL: <http://www.informatik.hu-berlin.de/top/pnml/about.html>.

Adresa

Ing. Petr Jedlička, Ph.D., Ústav informatiky, Mendelova zemědělská a lesnická univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika