Simulácia firmy prostredníctvom autonómnych agentov

Branislav Bošanský, Cyril Brom

Matematicko-fyzikálna fakulta, Univerzita Karlova v Prahe,
Kabinet software a výuky informatiky, Malostranské nám. 25, Praha 1
bbosansky@zoznam.sk, brom@ksvi.mff.cuni.cz

Abstrakt *Popis spoločnosti pomocou procesov je bežnou praxou, ktorá poskytuje manažérom či vedúcim pracovníkom prehľad o fungovaní firmy, pričom jej výhodou je predovšetkým jednoduchosť a zrozumiteľnosť výsledného návrhu. Na druhú stranu je však uvedená špecifikácia menej vhodná pre účely simulácie takto definovaných organizácií, kde v súčasnosti používané programy sú založené na štatistických vyhodnoteniach bez možnosti vizualizácie jej vlastného priebehu. Vhodnejšou metódou pre vytvorenie vierohodnej simulácie je preto využitie autonómnych agentov, poskytujúcich reálnejší obraz modelovanej oblasti s možnosťou vizuálnej prezentácie či interaktívnych zásahov zo strany užívateľa. Problémom tohto prístupu je však vo všeobecnosti vyššia náročnosť tvorby návrhu, vzhľadom k nutnosti špecifikovať akcie pre jednotlivých agentov. V článku sa preto venujeme spojeniu tejto oblasti agentov a ich využitia pri simulácii firmy definovanej pomocou procesov, kde pri riešení vychádzame z procesov riadených udalosťami (EPC z angl. Event-Driven Process Chains) a ich formálneho popisu EPML (EPC Markup Language), ktorý rozširujeme o ďalšie prvky nutné pre zachytenie informácií umožňujúcich „preloženie" tohto jazyka do podoby pravidiel pre reaktívnych agentov, interagujúcich v multiagentovom systéme, ktorý reprezentuje virtuálnu firmu. Článok popisuje aktuálny stav prebiehajúcej práce, kde špecifikujeme potrebné rozšírenia EPML a popisujeme ich interpretáciu a využitie vo výslednom multiagentovom systéme.*

1 Úvod

Pre študentov ekonomických a manažerských oborov by bolo vhodným obohatením výuky mať možnosť vytvorenia návrhu firmy a následne s ním interaktívne pracovať v podobe simulácie, kedy by sa im vizuálne prezentoval príslušný virtuálny svet, v rámci ktorého by virtuálni zamestnanci na základe daného návrhu plnili svoje úlohy. Pre uskutočnenie práve uvedeného, ale aj iných podobných zámerov, je tak potrebné preskúmať nielen možnosti definovania návrhu firmy, kde sa v praxi využíva najmä modelovanie pomocou procesov, ale aj možnosti jeho transformácie do podoby výslednej simulácie.

1.1 Procesné modelovanie

Procesné modelovanie (BPM z angl. *Business Process Modelling*) sa v priebehu 90-tych rokov stalo používanou technikou pre zachytenie organizácie práce v spoločnostiach, ich oddeleniach až po znázornenie jednotlivých pracovných postupov a metodík. Významnými hľadiskami, ktoré sa zaslúžili o jeho popularitu, sú:

- *prirodzenosť* - uvažovanie o práci resp. pracovných postupoch vo forme navzájom nadväzujúcich procesov je prirodzené (pracovník A pracuje na úlohe U1, keď skončí, pracovník B bude s výsledkom úlohy U1 pracovať na úlohe U2)
- *jednoduchosť* - zachytenie pracovného postupu do formy procesov nie je príliš zložité[1]
- *prehľadnosť a zrozumiteľnosť* - na základe procesných špecifikácií je možné veľmi rýchlo získať prehľad a zorientovať sa vo fungovaní firmy či tímu ľudí

Existuje niekoľko jazykov, prostredníctvom ktorých môžeme procesy firmy zachytiť. Všetky majú istú formu grafickej reprezentácie a ich návrh prebieha väčšinou opäť grafickou cestou. Za najvýznamnejšie spomeňme najmä jazyky UML, ktorý okrem procesného návrhu poskytuje oveľa širšie možnosti využívané najmä v rámci softwareového inžinierstva, BPEL používaný v oblasti popisu práce webových služieb či EPC [1] určený na procesnú špecifikáciu firmy, zachytenie hierarchie procesov a organizačnej štruktúry. Dôležitosť procesnej špecifikácie je zdôraznená aj jej nevyhnutnosťou pre splnenie medzinárodných noriem, napríklad noriem kvality za účelom získania certifikátov ISO 9001 [2].

1.2 Simulácia firmy a využitie agentov

Špecifikácia firmy pomocou procesov prináša výhody vo forme pochopenia organizácie práce a dáva priestor na jej zlepšenie, no môže taktiež poslúžiť ako základ pre ďalšie využite. Medzi také môžeme napríklad zaradiť systémy pre kontrolu plnenia procesov, systémy pre podporu v rozhodovaní pre manažérov a najmä, už spomínané, možnosti simulácie takto navrhnutých postupov, ktoré, ako sme už na začiatku predznamenali, sú pre nás kľúčovou oblasťou, pričom jej využitie nie je obmedzené len na výuku študentov. Možnosť vyskúšať si niekoľko rôznych variant riešení zadanej úlohy a

[1] myslené relatívne v porovnaní s inými formami popisu, ktoré budú spomenuté ďalej

vidieť tak ich dopad na modelovú situáciu, je využiteľné i pre reálne spoločnosti, no je však vždy nutné dbať na správnu interpretáciu výsledkov simulácie a na vloženie korektných parametrov či zvolení vhodnej úrovne jej detailnosti. Súčasné možnosti simulácií firiem vieme rozdeliť na dva hlavné prúdy:

1. pravdepodobnostné vyhodnotenie jednotlivých variant pracovných postupov a z nich vyplývajúce štatistiky hodnotiace napr. odhadovaný zisk
2. simulácie využívajúce agentový prístup, kde je firma reprezentovaná multiagentovým systémom

Reprezentantom prvého smeru je napríklad nástroj ARIS Toolset [3] popisujúci procesy prostredníctvom jazyka EPC. Po doplnení údajov o dĺžke trvania jednotlivých činností a priradení pracovníkov (pochopiteľne, jeho možnosti sú ďaleko väčšie) vyhodnotí prechody daných procesov pričom ako výsledok poskytne údaje o odhadovaných ziskoch, trvaní jednotlivých procesov a podobne. Bohužiaľ, takáto simulácia má základné nedostatky práve v podobe čisto pravdepodobnostného vyhodnotenia, pomocou ktorého nie je napríklad možné riešiť pracovné činnosti súvisiace s komunikáciou medzi jednotlivými aktérmi [4] či iné komplexnejšie problémy a často, ako napríklad v uvedenom prípade programu ARIS, chýba vizuálne znázornenie samotného priebehu simulácie. Pochopiteľne, je možné rozšíriť túto formu o niektoré používané simulačné techniky,[2] avšak neodstránime tak všetky spomenuté nedostatky.

Smer, naznačený ako druhý, je zaujímavý najmä z pohľadu distribuovanej umelej inteligencie. Ako uvádzajú autori v [5], procesy v reálnych spoločnostiach je ťažké aproximovať jednoduchou funkciou, na základe ktorej prebehne výpočet spomenutý v predchádzajúcom odstavci. Je to najmä z dôvodu ich nepredikovateľnosti, kooperácie ľudí v rámci tímu či naopak distribúcie práce a využití zdrojov. Tým, že firmu reprezentujeme ako multiagentový systém a jednotlivé procesy ako problémové úlohy pre agentov predstavujúcich jednotlivých pracovníkov, môžeme využiť ich pozitívne vlastnosti, ako reaktívnosť, proaktívnosť či schopnosť sociálneho správania. Príkladom využitia agentov pri implementácii business procesov je, už odkazovaná práca [5] modelujúca spoločnosť British Telecom. Ako je vidieť aj na príkladoch uvedených v odkazovanom článku, v prípade popisu firmy, respektíve aj všeobecne multiagentového systému, autori špecifikujú akcie jednotlivo pre každého agenta, pričom túto metódu môžeme označiť za prirodzenú (vzhľadom k tomu, že definujeme chovanie autonómnych entít) a je využitá aj napríklad pri agentových simulačných nástrojoch SeSAm [6] či Brahms [7].

[2] napríklad diskrétnu simuláciu

Ak sa skúsime zamyslieť nad možnou aplikáciou tejto metódy (napríklad v podobe *if-then* pravidiel) na popis reálnych spoločností zistíme, že formalizácia komplexných firiem vyžaduje vyššie znalosti (hlavne z oblasti informatiky a matematiky), nie je dostatočne prehľadná a ťažšie sa v nej orientuje prípadne hľadajú nedostatky. Práve rozdielnosť špecifikácie práce agentov oproti zaužívanému procesnému návrhu preto identifikujeme ako hlavnú prekážku ich väčšieho využitia pre možnosti simulácie firiem. Cieľom výskumu je preto:

1. návrh takého formálneho popisu, ktorý by kombinoval výhody procesnej špecifikácie a zároveň by bol schopný niesť informácie potrebné pre definovanie pravidiel pre agentov
2. vytvorenie a implementácia modelu virtuálnej firmy navrhnutej procesným modelovaním.

V tomto texte sa zameriavame prvú časť tohoto cieľa, pričom predstavujeme základné myšlienky prezentujúce aktuálny stav pokračujúceho výskumu[3]. Zvyšok článku je organizovaný nasledujúcim spôsobom: v druhej časti definujeme problém na nižšej úrovni, naznačíme možné spôsoby riešenia a zadefinujeme EPC. V tretej časti sa budeme ďalej venovať jazyku EPML a jeho rozšíreniam, ktoré sú nutné pre umožnenie prekladu do výsledného multiagentového systému. V posledných sekciách popíšeme spôsoby implementácie a taktiež ďalšiu prácu, a to ako myšlienky riešenia druhej časti výskumu, tak aj možnosti využitia v iných oblastiach.

2 Analýza problému

Ako sme uviedli v predchádzajúcej sekcii, cieľom práce je definovať jazyk, založený na procesnom princípe, ktorý by sme vedeli transformovať do podoby používanej na popis správania sa agentov v multiagentovom systéme, čím dokážeme zlúčiť už spomínané výhody procesného návrhu (jednoduchosť, prehľadnosť) s výhodami a silou agentov v oblasti simulácií. Pri definovaní takéhoto jazyka máme v zásade dve možnosti: buď vytvoriť nový jazyk, ktorý bude spĺňovať očakávané predpoklady, alebo za týmto účelom upraviť niektorý z existujúcich jazykov. Na začiatku práce sme po úvahe prvú možnosť zamietli z týchto príčin:

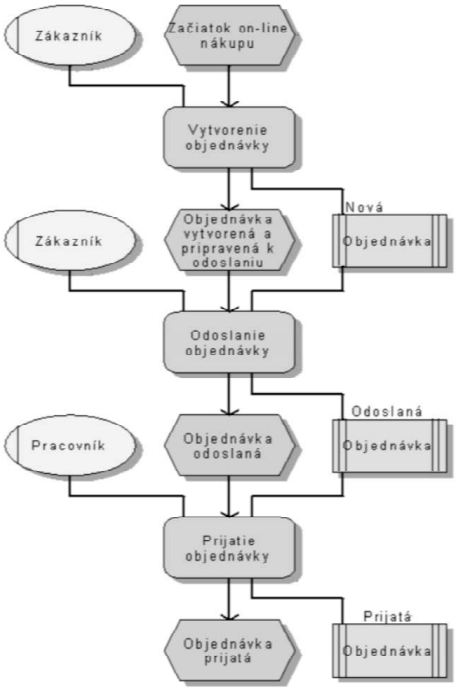
kompatibilita - bolo by výrazne náročnejšie dosiahnuť kompatibilitu s existujúcimi programami pre tvorbu procesných návrhov firiem

jednoduchosť používania - užívatelia by boli nútení pracovať s neznámym formalizmom, ktorý by museli naštudovať a osvojiť si ho

[3] momentálne prebiehajúceho vo forme diplomovej práce na MFF UK

využiteľnosť už uložených dát - vzhľadom na možné problémy s kompatibilitou s existujúcimi jazykmi, bolo by oveľa ťažšie využiť množstvo už uložených znalostí

Z uvedených dôvodov sme sa primárne venovali variante rozšírenia niektorého zo v súčasnosti používaných jazykov (UML, BPEL, EPC), spomínaných už v úvode, pričom ako základ bol nakoniec zvolený EPC. Dôvodom pre výber tohoto jazyka bola najmä jeho rozšírenosť a popularita (vychádza z neho ARIS, ale používa ho aj Microsoft Visio), zameranie práve na popis procesov firiem (aj v spojení s organizačnou hierarchiou) a vyššia výpovedná hodnota oproti napr. jazyku UML. Na druhú stranu, možnou komplikáciou vyplývajúcou z tejto voľby je nejednoznačnosť v definíciách a formalizovaných podobách EPC.



Obrázok 1. Príklad procesnej špecifikácie nákupu v online obchode znázornená pomocou EPC

2.1 Definícia EPC

Neformálne[4] môžeme EPC charakterizovať ako postupnosť procesov (v rámci EPC sú nazývané *funkcie*) doplnených o udalosti, prostredníctvom ktorých dochádza buď k aktivácii bezprostredne nasledujúceho procesu, resp. dôjde k výskytu danej udalosti po úspešnom ukončení procesu, ktorý jej predchádzal. Príklad jednoduchého EPC diagramu znázorňuje obrázok 1, kde vidíme udalosti znázornené ako zaoblené obdĺžniky a funkcie ako šesťuholníky. Pre rozdelenie kontrolného toku (z angl. *control flow*) využíva EPC tri konektory (*and, or* a *xor*), pomocou ktorých je možné vytvoriť niekoľko nezávislých tokov či ich opäť spolu synchronizovať.

Pod označením EPC však rozumieme aj ďalšie varianty a rozšírenia uvedenej základnej definície. Bežne využívané (napr. v programe ARIS, prípadne popísané v [9]) je doplnenie o nasledujúce elementy:

- *dátové pole* (*data field*) reprezentujúce dátové elementy (zdroje potrebné pre vykonanie procesu)
- *aktér* (*participant*), prostredníctvom ktorého môžeme zachytiť organizačnú štruktúru spoločnosti resp. pri spojení s procesom tak určujeme jeho vykonávateľa

Pochopiteľne existuje ešte niekoľko ďalších možných rozšírení (napr. v ARIS-e sú to špeciálne typy dátových polí ako *počítačový systém, telefón, fax* apod.), avšak pre náš zámer nie sú tieto potrebné. V rámci simulácie firmy potrebujeme totiž reprezentovať všetky entity, ktoré sa v rámci procesov využívajú a nie je potrebná žiadna špecializácia pre vybrané typy.

2.2 Formálny zápis EPC

Nejednoznačnosť v definícii ovplyvnila aj nejednoznačnosť vo formálnych zápisoch EPC, kde každý program využíva svoj vlastný formát [9]. Keďže, ako sme naznačili v predchádzajúcich častiach, budeme rozširovať možnosti jazyka EPC o uloženie informácií pre vytvorenie agentov, rozumnou požiadavkou pre formalizmus, z ktorého budeme vychádzať, je jednoduchosť, prehľadnosť a v ideálnom prípade dostatočná abstrakcia, ktorá by zahrňovala všetky ostatné formalizmy. Týmto požiadavkám takmer presne odpovedá jazyk EPML (z angl. *EPC Markup Language*) [9], ktorý vznikol práve za účelom premostenia jednotlivých formátov EPC.

3 Jazyk EPML a jeho rozšírenie

Jazyk EPML je založený na báze XML a zjednodušene teraz popíšeme jeho najdôležitejšie elementy (pre úplnú špecifikáciu viď [9] prípadne [10], príklad jazyka EPML, už doplneného o rozširujúce prvky, je zobrazený na obrázku 2). Základným prvkom EPML je stromová štruktúra jednotlivých EPC diagramov, kde jej

[4] formálnu definíciu a úplný popis jazyka je možné nájsť v [1], prípadne [8]

listom (teda samotným diagramom) je element ⟨*epc*⟩ a vnútorné uzly sú adresáre ⟨*directory*⟩. V EPC diagrame je reťazec procesov uložený pomocou tzv. *EPC elementov* reprezentujúcich udalosti ⟨*event*⟩, procesy ⟨*function*⟩, aktérov ⟨*participant*⟩, dátové položky ⟨*dataField*⟩ a konektory (⟨*and*⟩, ⟨*or*⟩ a ⟨*xor*⟩). Tie sú navzájom prepojené dvoma typmi orientovaných hrán – jedná sa o hrany kontrolného toku ⟨*arc*⟩ (spájajúce udalosti, procesy a konektory) a hrany relácií ⟨*relation*⟩ (spájajúce aktérov a dátové položky s procesmi).

Okrem vlastných EPC diagramov obsahuje štruktúra súboru EPML taktiež definície použitých atribútov (element ⟨*attributeTypes*⟩) a pomocou elementu ⟨*definitions*⟩ umožňuje vyčlenenie opakujúcich sa údajov jednotlivých objektov s cieľom sprehľadniť výslednú štruktúru. Príklad vidíme práve na obrázku 2 na objekte objednávky (element ⟨*dataField*⟩), ktorý sa pomocou atribútu *defRef* odkazuje na definíciu uvedenú na začiatku súboru.

3.1 Procesy z hľadiska simulácie

V nasledujúcej sekcii budeme hľadať informácie, ktoré musíme do klasického EPC doplniť, aby sme vedeli vytvoriť pravidlá pre jednotlivých agentov. Uvážme najprv základné požiadavky na proces a následne aj rozšírenia potrebné pre výslednú simuláciu:

vstupy - vstupné objekty na základe používanej varianty EPC môžu byť dvoch typov, a to buď informácia o aktivácií procesu v danom reťazci procesov (tj. prítomnosť kontrolného toku prostredníctvom predchádzajúcich elementov (funkcií, udalostí či konektorov)), alebo dátové objekty potrebné pre vykonanie daného procesu. Pre vstupné procesy potrebujeme navyše poznať dve kľúčové hodnoty atribútov - ich povinnosť (či sú pre vykonávanie tohoto procesu povinné) a ich využívanie počas trvania procesu. Úroveň využívania objektu je nutná pre možnosť simulovania procesov na hrubšej úrovni detailov, kedy v rámci priebehu dlhšieho procesu nie je tento objekt plne využívaný.

výstupy - medzi výstupné objekty zaraďujeme opäť predanie informácie o kontrolnom toku do nasledujúcich elementov a nastavenie výstupných hodnôt príslušných atribútov výstupných dátových objektov. Ďalej však ešte potrebujeme vedieť či má vzniknúť nový výstupný objekt, alebo sa má modifikovať hodnota objektu, ktorý bol pri priebehu procesu použitý a teda je nutné, aby v tomto druhom prípade niesol výstupný objekt odkaz na príslušný vstup

aktéri - aktéri sú prepojený s daným procesom a určujú, ktorí agenti sa majú na vykonávaní tohoto procesu podieľať. Podobne ako pre vstupné objekty u nich potrebujeme poznať ich povinnosť a úroveň využitia.

miesto - keďže je cieľovou platformou určitý virtuálny svet, agenti budú stelesnený a budú sa v ňom pohybovať, môžeme mať určené miesto, kde sa má tento proces vykonávať. Jeho určenie bude relatívne vzhľadom na určitý objekt.

priebeh procesu - ak chceme simulovať proces, potrebujeme pre jeho úspešné dokončenie poznať dĺžku jeho trvania. Tá by štandardne mala byť zadaná ako parameter pre proces, pričom v samotnom priebehu simulácie by jej skutočná hodnota bola modifikovaná v závislosti na vhodnej pravdepodobnostnej distribúcii. Ak však zoberieme do úvahy aj možnosť prerušenia vykonávania takéhoto procesu (čo je vzhľadom na zamýšľané použitie rozumná požiadavka), potrebujeme ďalej poznať jeho prioritu v porovnaní s ostatnými procesmi, ale hlavne celkovú priebehovú funkciu procesu, aby bolo možné u atribútov výstupných objektov nastaviť hodnoty odpovedajúce danému časovému okamžiku. Pre úplnosť je nutné poznamenať, že priebehová funkcia istých procesov sa vo všeobecnosti môže líšiť v závislosti od využitia voliteľných vstupných objektov, kooperácie viacerých agentov či miesta vykonávania procesu.

Uvedené rozšírenia nám poskytnú dostatok informácií pre simuláciu procesov, pričom vďaka určitej forme abstrakcie (hlavne v súvislosti priebehovou funkciou) sme schopný simulovať aj netriviálne procesy a nie je nutné vytvárať procesný návrh až do úplných detailov. Na druhú stranu rozšírení, ktoré sme momentálne do návrhu nezahrnuli, by bolo možné nájsť ešte niekoľko (napríklad definovanie úrovne vyťaženia vstupného objektu tiež pomocou funkcie alebo definovanie zložitejších logických podmienok nad vstupmi procesu) a určujú smer, ktorým by bolo možné ďalej prácu rozvíjať.

3.2 Reprezentácia rozšírených procesov

Ako už bolo naznačené, väčšina rozšírení je zastúpená doplnením jednoduchých atribútov (tj. reálne číslo, logická hodnota a pod.), čo je vďaka pravidlám EPML umožňujúcim do EPC elementov (ako *function*, *event*, atď.) vložiť element *attribute*, ľahko realizovateľná úloha. Na obrázku 2 vidíme príklad rozšíreného jazyka EPML, kde je zobrazená časť súboru popisujúceho proces z obrázku 1, konkrétne je vybraná funkcia "Odoslanie objednávky" spolu so súvisiacimi udalosťami, dátami a aktérom. Príklad ukazuje jednak základné elementy jazyka EPML ako sme ich v krátkosti špecifikovali na začiatku sekcie 3, ale taktiež vidíme doplnené atribúty pri jednotlivých vstupných objektoch

(ako sú objednávka – *dataField* a zákazník – *participant*) a taktiež pri výstupnom objekte (opäť objednávka reprezentovaná elementom *dataField* doplnená o odkaz na existujúci vstupný objekt).

Problematickou časťou rozšírenia je tak priebehová funkcia, ktorej uloženie v podobe XML je vzhľadom k možnému vysokému počtu dimenzií a parametrov (závisí na všetkých ostatných vlastnostiach procesu) v praxi nepoužiteľné. Zavádzame preto do EPML nový element $\langle transitionFunction \rangle$, ktorého predkom je *function* a ktorý odkazuje na triedu vyššieho jazyka[5] implementujúcu priebeh daného procesu, pričom v jazyku XML ponechávame jej nastavenia. V príklade na obrázku 2 tak vidíme použitie jednoduchej prechodovej funkcie, ktorá reprezentuje Heavsidovu skokovú funkciu. Návrh rozšírenia pre prechodové funkcie bol do prekladača jazyka EPML implementovaný tak, že spracovanie XML elementov vnorených v *transitionFunction* zabezpečuje samotná trieda reprezentujúca túto funkciu. Dôvodom pre toto rozhodnutie je vysoká variabilita výstupných dimenzií rôznych funkcií, ich parametrov a vplyvov týchto parametrov na priebeh funkcií jednotlivých hodnôt výstupných objektov. Vďaka tomuto oddeleniu tak môžeme pre popis priebehu procesu použiť ľubovoľnú funkciu, keďže nebudeme obmedzený možnosťami jej nastavenia.

Popíšme teraz elementy určujúce vlastnosti funkcie odpovedajúcej príkladu na obrázku 2, pričom podobné prvky sa nachádzajú aj v iných funkciách, ktoré budú v druhej fáze výskumu implementované. Kľúčovými prvkami pre proces sú nastavenie dĺžky jeho trvania, určenie konečných hodnôt (a ich možnej variability) u atribútov výstupných objektov a taktiež nastavenie parametrov funkcie, reprezentujúcej priebeh procesu. Väčšina z uvedených vlastností je v našom príklade zachytená elementom $\langle outputObjects \rangle$, kde v elemente $\langle expectedFinishTime \rangle$ zaznamenávame očakávanú dĺžku procesu a pomocou elementov $\langle outputDataObject \rangle$ nastavujeme hodnoty atribútov výstupných objektov. Oba druhy nastavení je možné doplniť ďalšími parametrami (pomocou elementu $\langle parameters \rangle$). Tie môžu jednak určovať mieru variability výstupnej hodnoty (parameter *divFunction* ukazujúci na pravdepodobnostnú distribúciu a jej parameter *diversity* reprezentujúci rozptyl), ale v prípade komplikovanejších funkcií pomocou nich taktiež určujeme vlastný priebeh výstupných hodnôt (napríklad v prípade sigmoidy jej strmosť). Opakujúce sa parametre je možné z dôvodu zjednodušenia uviesť na začiatku popisu prechodovej funkcie v elemente $\langle defaultParameters \rangle$.

Vzhľadom k tomu, že nutnosť programovania priebehových funkcií procesov pri ich návrhu nie je žiadúca, budú implementované triedy popisujúce jednodu-

[5] pre implementáciu je využívaná jazyk Java

```
<epml>
...
<attributeTypes>
   <attributeType typeId="orderState" />
   <attributeType typeId="zakaznikID" />
   <attributeType typeId="utilization" />
   <attributeType typeId="necessity" />
   <attributeType typeId="existingRef" />
   <attributeType typeId="priority" />
   <attributeType typeId="location" />
   ...
</attributeTypes>

<definitions>
   <definition defID="0003">
      <name>Objednavka</name>
      <attribute typeRef="zakaznikID" value="Zakaznik"/>
   </definition>
   ...
</definitions>

<directory name="Root">
<epc epcId="1" name="Online Shopping">
 ...
<event id ="3">
   <name>
     Objednavka vytvorena a pripravena k odoslaniu
   </name>
</event>

<arc id ="102">
   <flow source ="3" target ="4"/>
</arc>

<dataField id="32" defRef="0003" >
   <attribute typeRef="orderState" value="Nova" />
   <attribute typeRef="utilization" value="1.0" />
   <attribute typeRef="necessity" value="true" />
</dataField>
<relation id="73" from="32" to="4" />

<participant id="21">
   <name>Zakaznik</name>
   <attribute typeRef="utilization" value="1.0" />
   <attribute typeRef="necessity" value="true" />
</participant>
<relation id="74" from="21" to="4" />

<function id ="4">
   <name>Odoslanie objednavky</name>
   <attributeType typeRef="priority" value="10" />
   <transitionFunction className="cz.simphi.functions.Heaviside">
      <defaultParameters>
         <parameter type="divFunction"
            value="cz.simphi.distributions.GaussDistribution" />
      </defaultParameters>
      <outputObjects>
         <expectedFinishTime value="5">
            <parameter type="diversity" value="1" />
         </expectedFinishTime>
         <outputDataObject idRef="31">
            <attribute typeRef="orderState" value="Odoslana" >
               <parameter type="diversity" value="0" />
            </attribute>
         </outputDataObject>
      </outputObjects>
   </transitionFunction>
</function >

<dataField id="33" defRef="0003">
   <attribute typeRef="existingRef" value="32" />
</dataField>
<relation id="75" from="4" to="33" />

...
</epc>
</directory>
</epml>
```

Obrázok 2. Príklad procesu v rozšírenej EPML notácii

ché funkcie (ako napríklad lineárnu, skokovú, sigmoidu a pod.) a taktiež triedy reprezentujúce zložitejšie funkcie, kde príkladom môže byť zloženie výslednej funkcie z funkcií jednoduchých (t.j. funkcie sa budú líšiť pre jednotlivé výstupné objekty a ich atribúty).

Dôležitým predpokladom pre spracovanie takto rozšírených procesov je nutná ich plná inštanciovanosť (tj. musia byť zadané všetky vymenované atribúty). Tento fakt je možné po teoretickej stránke v rámci EPML vyriešiť najmä za využitia elementu *definitions*, praktická implementácia však bude riešená až pri tvorbe samotného nástroja, ktorý však spadá mimo rozsah nášho výskumu.

3.3 Prevod rozšírených procesov na pravidlá

Uveďme teraz interpretáciu jednoduchého príkladu z obrázku 2 v podobe pravidiel a priebehu jeho simulácie. V popise správania sa agenta reprezentujúceho zákazníka *Zákazník*, by sa tak v predpokladoch pravidla nachádzala podmienka na výskyt udalosti *Objednávka vytvorená a pripravená k odoslaniu*, dostupnosť agenta (teda agent nevykonáva inú činnosť s vyššou prioritou) a na existenciu objednávky tohoto zákazníka v stave *Nová*. Po splnení podmienok by bolo prostredníctvom priebehovej funkcie spustené vykonávanie procesu týmto agentom a po jeho skončení (vzhľadom k použitej skokovej funkcii) by došlo k nastaveniu stavu použitej objednávky na *Odoslaná* a k vyvolaniu udalosti *Objednávka odoslaná*.

Pochopiteľne, uvedený príklad bol ukážkou jednoduchej transformácie a pri implementácii končeného virtuálneho sveta bude realizovaný preklad komplexnejších procesov (napr. procesov s rozčlenenými tokmi, variabilnejšími vstupnými a výstupnými objektmi a takisto hierarchických procesov).

4 Implementácia a nadväzujúca práca

Implementácia, rovnako ako aj ciele samotnej práce, je rozdelená na dve časti. Prvá odpovedá vytvoreniu prekladača rozšíreného jazyka EPML do podoby všeobecných pravidiel, kde pre ich validáciu využívame systém JBoss Rules[11] a v ďalšom priebehu bude nasledovať vytvorenie finálnej simulácie, ktorá bude odpovedať časti reálnej firmy. S tým súvisí namodelovanie odpovedajúceho virtuálneho sveta, vo vhodnom simulačnom multiagentovom systéme, pričom pravdepodobne bude využitý nástroj IVE [12].

5 Záver

V článku bolo predstavené zadanie a postup riešenia práce, ktorá si za cieľ kladie vytvoriť simuláciu firmy s využitím autonómnych agentov. V doterajšom priebehu riešenia, ktorý je pokrytý týmto textom, bol kladený dôraz na špecifikovanie vhodného jazyka pomocou ktorého by bolo možné tento cieľ naplniť.

Rozsahom tejto práce však naznačená problematika rozhodne nie je vyčerpaná. Drobné možnosti rozšírenia (za účelom zreálnenia simulácie) boli uvedené priamo v texte, no možnosti sú oveľa širšie. Je to z dôvodu, že procesný spôsob návrhu práce a fungovania multiagentového systému nie je používaný, no jeho uplatnenie by mohlo byť okrem simulácie firiem použiteľné aj napr. v oblasti formalizovaných lekárskych doporučení a preto bude výskum v tejto oblasti pokračovať.

Práca na tomto texte bola čiastočne podporená grantovým projektom GA UK ČR 351/2006/A-INF/MFF, grantom „Information Society“ pod číslom projektu 1ET100300517 a grantom Ministerstva Školstva ČR MSM0021620838. Poďakovanie patrí taktiež Petrovi Kocábovi zo spoločnosti SoftDeC s r.o.[6].

Referencie

1. G. Keller, M. Nüttgens, and A.-W. Scheer. *Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK)*. Technical Report Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Heft 89, Universität des Saarlandes, January 1992.
2. Int. Organisation for Standardisation: ISO 9001–4, Quality Systems. `http://www.iso.org`
3. IDS Scheer AG: ARIS Design Platform, ARIS Simulation. `http://www.ids-scheer.com`
4. Sierhuis, M. *Modeling and Simulating Work Practice*, Universiteit van Amsterdam. 2001
5. Jennings, N. R.; Faratin, P.; Norman, T. J.; O'Brien, P. and Odgers, B. Autonomous Agents for Business Process Management *Int. Journal of Applied Artificial Intelligence,* vol. **14**, pp. 145–189, 2000
6. Klügl, F.; Herrler, R. and Fehle, M. SeSAm: Implementation of Agent-Based Simulation Using Visual Programming. In *Proceedings of the AAMAS 2006*, 2006
7. W. J. Clancey, P. Sachs, M. Sierhuis, and R. van Hoof. Brahms: Simulating practice for work systems design. *International Journal of Human-Computer Studies*, vol. **49**, pp. 831—865, 1998.
8. M. Rosemann and W. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, vol. **32**, pp. 1–23, 2007
9. Mendling, J. and Nüttgens, M. *EPC Markup Language*, Technical Report, Vienna University of Economics and Business Administration, 2005
10. Mendling, J. EPML XML Schema, `http://wi.wu-wien.ac.at/home/mendling/EPML/EPML_12.xsd`, Version 1.2
11. JBoss Rules, Version 3.0.6, `http://www.drools.org`
12. IVE, stránky projektu, `http://mff.modry.cz/ive/`

[6] http://www.softdec.cz