PROGRAMAREA ORIENTATĂ SPRE AGENŢI: LIMBAJE DE PROGRAMARE, INSTRUMENTE ŞI PLATFORME

Inga LISNIC, Sergiu SCROB

Universitatea Tehnică a Moldovei

Abstract: *În lucrarea dată este prezentată o imagine de ansamblu asupra dezvoltării rapide a agenților inteligenți și în special sunt prezentate limbaje de programare, instrumente și platforme utilizate la programarea agenților inteligenți. O atenție deosebită este acordată limbajelor semnificative concepute și proiectate pentru a sprijini implementarea sistemelor bazate pe agenți și prezentate aplicațiile lor pentru diferite domenii.*

Cuvinte cheie**:** *agent inteligent, programarea orientată spre agent, instrumente, platforme, sisteme bazate pe agenți.*

1. Noţiuni generale

Metafora "agenți inteligenți" ca element de bază pentru dezvoltarea noilor generații de sisteme inteligente de software a declanşat cercetări ştiinţifice teoretice şi experimentale, care vizează dezvoltarea de noi limbaje de programare pentru sistemele cu agenți inteligenți [1].

Există o serie largă de definiții a unui "agent inteligent", unde se include o proprietate comună şi anume agentul acționează în numele utilizatorului, şi posedă o mulțime de proprietăți suplimentare:

- agent comunică cu alți agenți într-un sistem cu mai mulți agenți;
- acționează în mod autonom;
- este inteligent;
- învață din experiență;
- acționează proactiv şi reactiv;
- este modelat şi / sau programat folosind caracteristici asemănătoare omului (credințe, intenții, scopuri, acțiuni etc.);
- este mobil ş.a.

După mai mult de două decenii de activitate ştiinţifică în domeniu, provocarea constă în includerea agenților în medii software reale şi folosirea pe scară largă a paradigmei agenților în programarea mainstream. O modalitate de a facilita acest lucru este de a furniza limbaje de programare orientate spre agent, instrumente şi platforme.

Cu cincisprezece ani în urmă dezvoltarea sistemelor de agenți şi multi-agenți a fost privită ca un domeniu în proces de dezvoltare şi cercetare şi care a cunoscut o dezvoltarea rapidă. Se poate de menționat, că principala realizare a acestei tendințe de dezvoltare şi cercetare a fost elaborarea de noi modele de programare, care abordează atât caracteristici de bază a agenților inteligenți (autonomie, reactivitate, proactivitate şi abilități sociale), cât şi caracteristici avansate, de "atitudini mentale" (credințe, dorințe, intenții, angajamente), urmând modelul "sistemelor intenționate" introduse de filosoful Daniel Dennett în 1971 pentru a explica comportamentul agenților raționali [2]. Tehnologii orientate spre agent, ingineria sistemelor de agenți inteligenți, limbajele de programare spre agent reprezintă o arie de cercetare activă şi emergentă în proces de dezvoltare.

Există multe abordări, teorii, limbaje, seturi de instrumente şi platforme de calitate utilizate în programarea sistemelor multi-agent. Scopul principal al acestei lucrări este de a prezenta un sondaj în domeniul tehnologiei agenților inteligenți, programării orientate spre agenți (AOP) şi sistemelor multi-agent (MAS).

2. Modele de programare orientat- Agent

Termenul Programare orientată spre agent (AOP) defineşte o nouă paradigmă de programare. Acesta reprezintă un cadru computațional a cărui noțiune compoziționlă centrală este un agent, văzută ca o componentă software cu calități mentale, abilități de comunicare şi o noțiune de timp. AOP este considerată ca fiind o specializare a programării orientate-obiect (OOP), dar există unele diferențieri importante între aceste concepte şi anume obiectele şi agenții diferă în gradul lor de autonomie.

Spre deosebire de obiectele care invocă direct acțiunile altor obiecte, agenții îşi exprimă dorința de a executa o acțiune. De asemenea, agenții pot avea deseori interese conflictuale, astfel încât ar putea fi dăunător pentru un agent să execute o cerere de acțiune de la un alt agent.

O diferență suplimentară este flexibilitatea, agenții manifestă adesea comportamente proactive și adaptive și folosesc învățarea pentru a-și îmbunătăți performanțele în timp, firul de control este diferența majoră finală, în timp ce sistemele multi-agent sunt deduse în mod implicit prin multi-agent, există de obicei un singur fir de control în OOP.

Paradigma AOP a dus la dezvoltare a limbajelor de programare spre agenți, astfel s-a dezvoltat o serie de limbaje care sunt prezentate în figura 1.

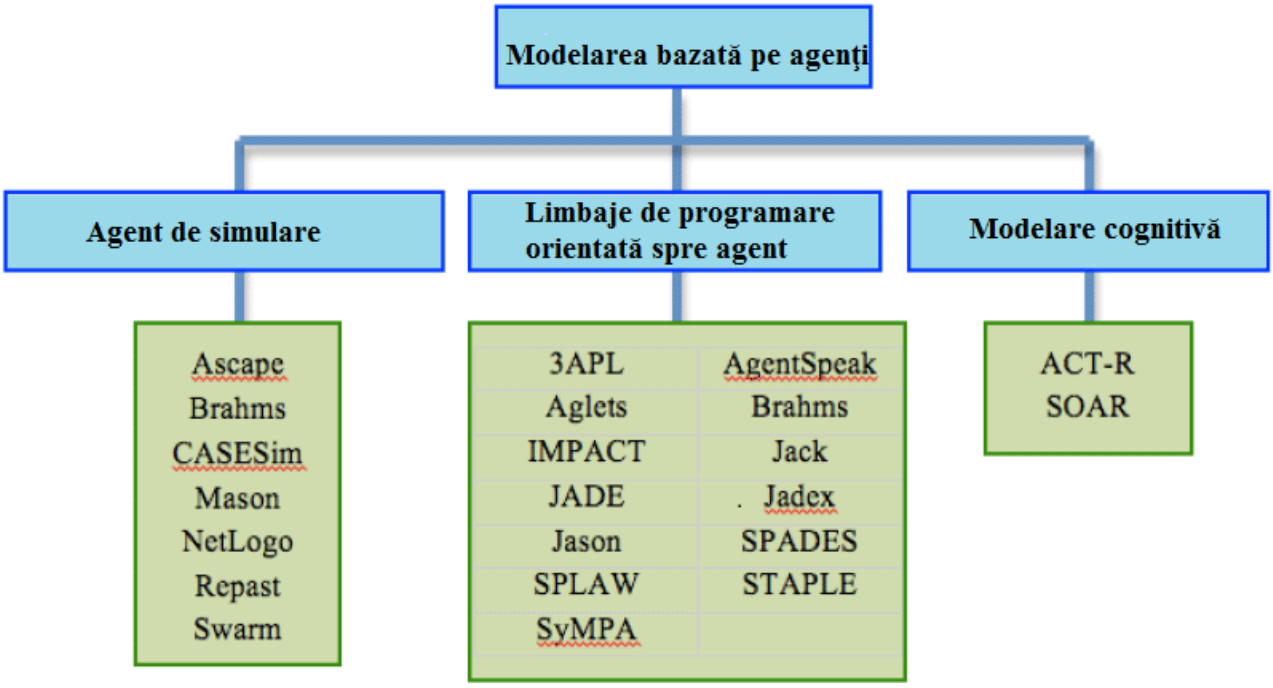

Fig. 1. Sumar specific limbajelor de programare spre agenţi.

3. Instrumente şi platforme

Un set de instrumente pentru agenți (figura 2) este o infrastructură software mai complexă, care permite dezvoltarea și implementarea unui sistem multi-agent.

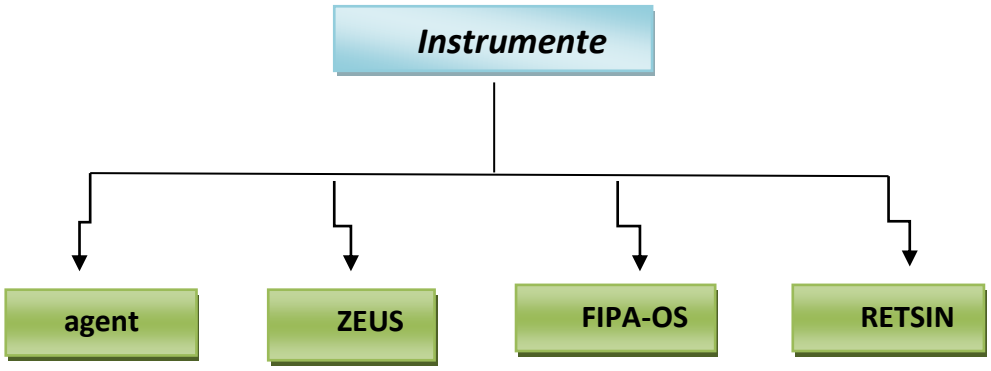

Fig. 2. Instrumente pentru agenţi.

AgentTool este un mediu / instrument de dezvoltare grafică bazat pe Java, care sprijină metodologia Multi-Agent Systems Engineering (MSE) inițial dezvoltată la Laboratorul de Inteligență Artificială a Institutului de Tehnologie al Forțelor Aeriene din Ohio. Implementează toate etapele MaSE, inclusiv conversația verificarea și generarea de coduri. Una dintre abilitățile sale cele mai interesante este posibilitatea de a lucra pe diferite părți ale sistemului și la diferite niveluri de abstractizare interschimbabil, care reflectă capacitatea MaSE de a adăuga treptat detalii.

ZEUS este unul dintre cele mai complete și mai puternice instrumente de agenți care sunt folosite pentru a proiecta, dezvolta și organiza sisteme de agenți. Scopul proiectului ZEUS a fost acela de a facilita

dezvoltarea rapidă a aplicațiilor multi-agent prin abstrarea într-un set de instrumente a principiilor și componentelor comune care stau la baza unor sisteme multi-agent existente.

FIPA-OS este un set de instrumente bazat pe componente, care permite dezvoltarea rapidă a agenților compatibili cu FIPA.

RETSINA este probabil unul dintre cele mai timpurii software-uri cele mai influente infrastructuri pentru dezvoltarea de sisteme multi-agent. Acesta susține dezvoltarea comunităților de agenți eterogeni care se pot angaja în relații de tip peer-to-peer fără a impune controlul centralizat pentru gestionarea agenților. Un sistem multi-agent bazat pe RETSINA este independent de platformă, fiind capabil să ruleze pe diferite sisteme de operare, în timp ce agenții săi pot fi implementate folosind diferite limbaje de programare cu scop general.

Platformele de agenți pot fi extrem de utile deoarece simplifică considerabil dezvoltarea și implementarea unui sistem multi-agent. Există opțiunea de a alege între platforme de agenți standardizate sau ne-standardizate. O platformă standard de agenți este compatibilă cu standardele disponibile pentru agenții de software. Respectarea standardelor este importantă pentru sistemele deschise, adică pentru sistemele care ar putea fi necesare în viitor pentru interoperabilitate cu alte sisteme care nu sunt disponibile în momentul dezvoltării sistemelor deschise sau că, chiar dacă acestea sunt disponibile în prezent, încă s-ar putea schimba în viitor. Conform literaturii [3], au fost dezvoltate peste 100 de limbaje specializate pentru programarea sistemelor multi-agent, însă doar câteva dintre ele în prezent sunt folosite pentru proiectarea sistemelor multi-agent.

În figura 3 sunt prezentate platformele pentru dezvoltarea agenților inteligenți.

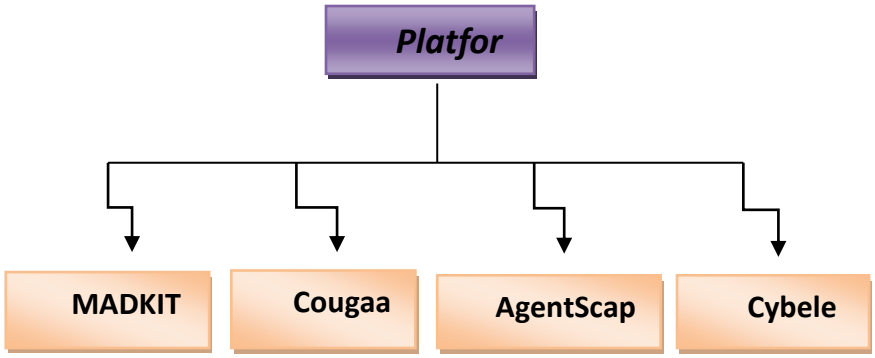

Fig. 3. Platforme de agenți.

Kitul de dezvoltare multi-agent - **MadKit** este o platformă multi-agent open source modulară și scalabilă, dezvoltată la LIRMM (Franța), construită pe baza modelului organizațional AGR (Agent / Grup / Role). MadKit este scris în Java și agenții MadKit joacă roluri în grupuri și astfel creează societăți artificiale. Pe lângă conceptele AGR, platforma adaugă trei principii de proiectare: arhitectura micro-kernel; agentificarea serviciilor; model grafic component. Ultima versiune a fost lansată în noiembrie 2010. MadKit este un set de pachete de clase Java care implementează kernel-ul agentului, diverse biblioteci de mesaje, sonde și agenți.

Cognitive Agent Architecture - Cougaar este o platformă open source bazată pe Java, dezvoltată ca rezultat al unui proiect multi-anual de cercetare DARPA. Cougaar nu este compatibil cu FIPA și, mai important, nu a fost conceput pentru respectarea standardelor. Agenții Cougaar sunt compuși din pluginuri care comunică distribuirea spațiului comun și distribuit de date – tablă arhitectură. Agenții se pot abona pentru primirea automată a actualizărilor tabloului de bord.

AgentScape este o platformă care în prezent suferă de problema că documentația, nu este destul de matură și este destul de incompletă. Cu toate acestea, AgentScape a fost aplicat într - o serie de proiecte interesante de cercetare și comerciale legate de piața energiei electrice și comerțul electronic.

CybeleTM este construit pe platforma Java. Agenții sunt programați în Java folosind un stil standard de programare numit Activity Centric Programming (ACP). Acest lucru înseamnă că blocurile de bază ale unui agent sunt activități, în timp ce accesul la funcționalitățile de bază ale CybeleTM este asigurat printr-o interfață de programare orientată pe activități (AOPI). CybeleTM permite dezvoltarea de aplicații distribuite prin instalarea acestora pe mai multe (cel puțin 2) noduri de rețea care împreună definesc o comunitate CybeleTM. Exact un singur nod este desemnat ca nod principal, în timp ce ceilalți sunt sclavi noduri. Un nod CybeleTM poate găzdui mai multe aplicații Java specializate cunoscute sub numele de containere

CybeleTM. Un container furnizează mediul de rulare pentru un set de agenți CybeleTM. Nu este greu de observat că o activitate în CybeleTM are asemănări cu comportamentul în JADE, precum și cu un plugin în Cougaar.

Concluzii

În ultimul deceniu, s-au făcut progrese esențiale în conceperea și dezvoltarea limbajelor de agenți de software și în implementarea sistemelor multi-agent. Cu toate acestea, aceste limbaje nu sunt imediat utile pentru dezvoltarea sistemelor reale, ci sunt utilizate mai degrabă pentru cercetare în înțelegerea sistemelor complexe folosind instrumente de modelare și simulare bazate pe agenți, ca limbaje de simulare a agenților.

Bibliografie

1. Mascardi, V.: Coo-BDI: Extending the BDI Model with Cooperativity, In Declarative Agent Languages and Technologies, Vol. 2990, pp.109-134 (2004)
2. Azarmi, N., Thompson, S.: "ZEUS: A Toolkit for Building Multi-Agent Systems", Proceedings of fifth annual Embracing Complexity Conference, Paris, (2000)
3. Bordini, R.H., Dix, J., Dastani, M., Seghrouchni, A.E.F.: Multi-Agent Programming Languages, Platforms and Applications, Springer, (2005)
4. Ferrein, A.: golog.lua: Towards a Non-Prolog Implementation of Golog for Embedded Systems. In: Cognitive Robotics, Dagstuhl Seminar Proceedings, no.10081, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Also in Proceedings of AAAI Spring Symposium 2010 on Embedded Reasoning, Stanford University, (2010)
5. Ross, R., Collier, R., O'Hare, G.: AF-APL: Bridging principles and practices in agent oriented languages. In Programming Multi-Agent Systems, Second Int. Workshop (ProMAS'04), volume 3346 of LNCS, Springer Verlag, pp. 66–88, (2005)
6. Nguyen G., Dang T.T, Hluchy L., Laclavik M., Balogh Z., Budinska I.: AGENT PLATFORM EVALUATION AND COMPARISON, Institute of Informatics, Slovak Akademy of Sciences (2002)
7. Sardina, S., Lespérance, Y.: Golog Speaks the BDI Language. In: 7th International Workshop on Programming Multi-Agent Systems, ProMAS 2009, LNCS 5919, pp.82-99, Springer (2010)
8. Thomas, R.S.: " Agent Oriented Programming Language", PhD thesis,Computer Science Department, Stanford University, Stanford, CA 94305, (1993)
9. Santoro, C.: Towards an Agent Programming Language, In 10th national workshop Towards the Future of Agent-based software systems, Parma, Italy (2009)
10. Clark, K. L., McCabe, F. G.: Go! – A Multi-paradigm Programming Language for Implementing Multi-threaded Agents, In Annals of Mathematics and Artificial Intelligence, Vol. 41, Issue 2 – 4, pp. 171 – 206, (2004)
11. Huang, J., Jennings, N., Fox, J.: "An Agent Architecture for Distributed Medical Care", Intelligent Agents, Lecture Notes in Artificial Intelligence, Vol 890, Springer-Verlag, pp. 219-232. (1994)