Explorarea Spatiului de Proiectare a Limbajelor Orientate pe Agent

Alex Muscar

November 19, 2013

Abstract

Atunci cand a fost introdusa de Shoham, aproape acum doua decenii, paradigma Agent Oriented Programming (AOP) a fost proiectata ca o alternativa de nivel inalt la Object Oriented Programming (OOP) pentru dezvoltarea sistemelor software complexe [15]. In urma efortului considerabil depus in ultimii 20 de ani pentru cercetarea paradigmei AOP, a fost propusa o gama larga de abordari pentru dezvoltarea unor astfel de sisteme. Dar promisiunile facute de proponetii timpurii ai paradigmei orientate pe agent nu s-au materializat, si aceasta nu a captata atentia comunitatii largi de dezvoltare a software-ului. In ciuda faptului ca promisiunile facute sunt tentante—productivitate sporita si scaderea barierei de intrare in domeniul dezvoltarii de software pentru incepatori—paradigma nu a reusit sa capteze atentia dezvoltatorilor de software. Agentii sunt considerati, in principal, unelte folosite pentru experimentarea in domeniul Artificial Intelligence (AI), nu unelte folositoare pentru dezvoltarea de aplicatii software practice.

Motivate de lipsa de succes a paradigmei orientate pe agenti, exista voci care ii prezic declinul [6]. In acelasi timp, unii membri ai comunitatii, incearca sa traga un semnal de alarma in legatura cu lipsa de coeziune si de teluri comune de care da dovada comunitatea AOP [4, 13].

Autorul este de parere ca lipsa de succes a paradigmei orientata pe agenti are printre cauze si esecul cercului de proponenti ai AOP de a oferi solutii pentru problemele cu care se confrunta industria dezvoltarii de software. Programarea concurenta este, de exemplu, un domeniu in care se pot aduce inbunatatiri substantiale. Prin inbunatatirea suportului pentru calcul concurent in limbajele orientate pe agenti, paradigma ar deveni utilizabila pentru dezvoltarea unei game variate de aplicatii, de la aplicatii stiintifice cum sunt simularile masiv paralele [17], la aplicatii financiare, asa cum este *high frequency trading* [7]. Este opinia noastra ca, prin tratarea unor aspecte care sunt mai apropiate de nevoile curente ale industriei software, comunitatea proponentilor AOP poate sa atraga atentia unei audiente mai largi.

Data fiind traditia mostenita de comunitatea AOP din domeniul AI, este de inteles de ce majoritatea eforturilor membrilor sai s-au indreptat catre dezvoltarea modelelor pentru *agenti rationali*, in timp ce alte aspecte ale sistemelor formate din agenti nu au primit atentia cuvenita. Exista o colectie vasta de modele formale pentru agenti si sisteme multi-agent, dar lucrurile sunt mult mai putin variate atunci cand vine vorba de unelte, si aici ne referim in mod special la Agent Oriented Programming Language (AOPL) [1]. Exista o dihotomie in domeniul uneltelor folosite pentru dezvoltarea aplicatiilor care folosesc agenti: in timp ce solutiile de nivel inalt se axeaza aproape exclusiv pe specificarea agentilor rationali, majoritatea uneltelor menite dezvoltarii de aplicatii industriale sunt implementate folosind tehnologii generice, care nu au foarte multe de oferit la capitolul expresivitate, e.g. JADE[1].

Mai exista, de asemenea, o problema "sociala" atunci cand discutam despre limba-

[1]Java Agent Development Framework: http://jade.tilab.com/

jele orientate pe agent: majoritatea se adreseaza unui grup restrans de indivizi cu nevoi specifice. De cele mai multe ori au o sintaxa greu de urmarit si nu ofera facilitati necesare dezvoltarii proiectelor complexe. Acest aspect le afecteaza si calitatea ca limbaje "educationale". Suntem de parere ca existenta unor limbaje orientate pe agent, care sa fie usor de inteles si folosit de cate studenti ar incuraja adoptarea paradigmei pe scara larga.

In utlimii ani situatia a inceput sa se schimbe, interesul pentru AOPL se afla pe o panta ascendenta. Dovada acestui fapt este cresterea numarului de propuneri pentru noi limbaje [5, 11]. De asemenea, o serie de aplicatii practice au fost implementate folosind limbaje AOP. Mai exact, limbajul Jason [2] a fost folosit pentru dezvoltrea unor aplicatii variate, pornind de la aplicatii web [10] si terminand cu aplicatii mobile [14].

Blueprint

Premiza acestei teze este ca inca este loc pentru o solutie de nivel inalt, care insa sa nu sacrifice aspectele pragmatice. Prin solutide nivel inalt intelegem un limbaj care reifica conceptele specifice domeniului, e.g. agenti, planuri si intentii. Scopul nostru este acela de a investiga aceasta abordare si implementarea sa. Programatorii pot fi atat de productivi pe cat le permit uneltele pe care le folosesc sa fie. Asadar, limbajele de progremare sunt elemente esentiale in dezvoltarea sistemelor software complexe, sistemele multi-agent facand parte din aceasta categorie. Aspectele discutate in aceasta teza vor fi tratate din perspectiva proiectarii si implementarii limbajelor de programare.

Principala contributie a acestei teze este proiectarea si implementarea limbajului Blueprint, un limbaj orientat pe agenti, care are ca tinta platforma Common Language Runtime (CLR). In proiectarea limbajului s-a pus accentul pe cateva aspecte importante: concurenta, verificarea la compilare a respectarii invariantilor specificati de programator, usurinta in utilizare si extensibilitatea. Fiecare dintre aceste aspecte adauga propria sa contributie in domeniile specifice din AOP si AOPL.

Inainte de a intra in detalii in legatura cu factorii care au motivat dezvoltarea limbajului BLUEPRINT si caracteristicile acestuia, trebuie mentionat faptul ca proiectarea sa a fost un proces organic. Germenul limbajului a fost un Embedded Domain Specific Language (EDSL) pentru dezvoltarea orientat pe agent, construit peste limbajul F#. Aceste idei s-au transformat, apoi, intr-o extensie adusa limbajului Jason, extensie al carei rol era acela de a facilita programarea concurenta. In sfarsit, ideea si-a realizat potentialul sub forma unui limbaj orientat pe agenti de sine statator.

Modelul pentru executia concurenta este o adaptare a celui prezentat in [16] adaptata pentru limbajele orientate pe agenti. Acest model este o descriere monadica a unitatilor de executie concurente, care permite o descriere naturala a compozitiei acestor unitati de executie. O contributie inrudita este utilizarea sabolnului *thread-pool* pentru executarea actiunilor care compun un plan. Aceasta abordare este in ton cu tendintele curente din comunitatea AOPL [3].

O a doua contributie importanta este proiectarea unui sistem de tipuri manifest pentru limbajul BLUEPRINT. Aceasta caracteristica a limbajului permite dezvoltarea unor aplicatii solide. Si acest aspect este in ton cu tendintele curente din comunitatea AOPL: de curand au inceput sa fie publicate eforturi de dezvoltare a unor limbaje orientate pe agenti cu sistem de tipuri manifest [12]. Si, ramanand in aria verificarii la compilare a invariantilor, BLUEPRINT adopta conceptul de *protocoale* pentru canalele de comunicare, care forteaza schimbul de mesaje prin intermediul canalelor de comunicatie sa respecte o schema definita de programator. Desi concepte similare exista si in alte limbaje, BLUEPRINT este primul limbaj cu aceasta caracteristica. Limbajul are, de asemenea, canale de comunicare asincrone si asimetrice, care au si rolul de puncte de sincronizare in programele agent.

Sistemul de tipuri manifest al limbajului BLUEPRINTsi protocoalele de comunicatie verificate la compilare impiedica o gama larga de erori care pot aparea pe parcursul proiectarii si implementarii unui proiect folosind solutiile AOPL curente, care au doar

sisteme de tipuri latente. Acest aspect face programele scrise in BLUEPRINT mai usor de extins si depanat. Astfel, utilizand BLUEPRINT, pot fi dezvoltate proiecte de mai mare anvergura. De asemenea, aceste trasaturi fac limbajul potrivit pentru utilizarea in activitatile educationale. Pentru a incuraja folosirea acestuia ca limbaj didactic, am decis sa folosim o sintaxa familiara programatorilor obisnuiti cu limbajele din familia C. Tot pentru a face limbajul mai usor de utilizat atat in scop industrial cat si didactic, acesta face utilizarea bibliotecilor scrise pentru platforma tinta usor de folosit.

Am comparat performanta implementarii limbajului BLUEPRINT cu cea a implementarilor principalelor surse de inspiratie in dezvoltarea acestuia: Jason si JADE. Asa cum era de asteptat, modelul de executie adoptat de limbajul propus de noi, e.g. planuri asincrone executate intr-un thread pool, il evidentiaza drept o solutie scalabila in comparatie cu JADE. Desi Jason se apropie de performanta de BLUEPRINT—si il si intrece in anumite conditii—are dezavantajul ca ramane fara memorie cand este pornit un numar mare de agenti. Experimentele facute de noi recomanda limbajul BLUEPRINT drept o solutie viabila pentru dezvoltarea sistemelor multi-agent, pornind de la cativa agenti si mergand pana la cateva milioane.

Decizia noastra de a studia executia concurenta in contextul AOPL a fost motivata de un proiect de cercetare mai amplu, care are ca scop dezvoltarea unui mecanism pentru negocieri dinamice [11, 13, 12]. Am luat decizia de a folosi limbajul Jason pentru implementarea prototipului initial, dar, data fiind natura distribuita a solutiei propuse, ne-am confruntat in curand cu limitarile limbajului Jason. Abordarea prezentata in aceasta teza este bazata pe rezultatele prezentate in [8, 9, 3, 5, 4]. Proiectarea limbajului BLUEPRINT este de asemenea bazata pe unele dintre observatiile prezentate in [1]. O analiza a utilizarii sablonului thread pool a fost prezentata in [10, 2]. In acest moment investigam o posibila extensie a modelului de concurenta adoptat de limbaj. O parte dintre rezultatele preliminare ale acestor eforturi au fost prezentate in [6].

Desi rezultatele prezentate in aceasta teza se adreseaza in mod direct scenariilor dis-

tribuite, aspectele concurente sunt prezente in mod inerent in astfel de sisteme. Asadar, tratarea acestei probleme, chiar si in contextul unui singur agent sau a mai multor agenti care ruleaza pe aceeasi masina, o sa influenteze pozitiv si eforturile viitoare in aceasta directie.

Studiul aspectelor legate de garantarea la compilare a invariantilor specificati de programator a mai fost influentata de un aspect in afara dorintei de a proiecta un limbaj care permite dezvoltarea sistemelor robuste. Acesta este experienta pe care am acumulat-o de-a lungul celor trei ani de doctorat in care am predat tehnologii orientate pe agent, mai exact JADE si Jason, studentilor de la licenta. Desi este scris in Java, si astfel beneficiaza de sistemul de tipuri manifest al acestuia, JADE nu ofera nici o garantie cand vine vorba de transmiterea de mesaje. Una dintre cele mai des intalnite probleme cu care s-au confruntat studentii atunci cand dezvoltau programe folosind JADE era ca trimiteau mesajele in ordinea gresita—nu respectau protocolul de comunicare—sfarsind astfel fie intr-o situatie de interblocare (deadlock), ori cu exceptii criptice aruncate de bilbioteca JADE din cauza nerespectarii ordinii mesajelor. Pe de alta parte, in urma predarii limbajului Jason la studentii de licenta, a fost intelegerea importantei pe care o are sintaxa limbajului in intelegerea unui limbaj nou. Majoritatea studnetilor, familiarizati doar cu limbaje din familia C, intampinau probleme cu sintaxa "exotica" a limbajului Jason. Bineinteles ca acesta este un aspect subiectiv, dar suntem de parere ca prin folosirea unei sintaxe cu cre studentii sunt familiarizati, o sa facem limbajul mai usor de invatat.

De asemenea schitam o posibila directie viitoare pentru dezvoltarea limbajului Blueprint: traducerea agentilor la limbajul JavaScript. Aceasta traducere deschide o serie de directii interesante. Suntem de parere ca domeniul aplicatiilor web are foarte mult potential cand vine vorba de AOP, pentru ca este o problema curenta care inca nu are o solutie satisfacatoare. Numarul foarte mare de limbaje de programare care au ca tinta browser-

ul sta dovada a fpatului ca aceasta nisa este inca deschisa[2]. Credem ca paradigma AOP este o solutie viabila pentu dezvoltarea aplicatiilor client (in browser), datorita naturii granulare si inerent comunicativa a agentilor. De asemenea, dezvoltarea aplicatiilor web poate fi o rampa de lansare in cresterea popularitatii paradigmei orientat pe agenti.

Un alt aspect benefic este faptul ca folosirea briwser-ului ca mediu de executie faciliteaza experimentarea cu limbajul, pentru ca un browser este prezent pe aproape orice computer. Acesta este o directie interesanta mai ales din perspectiva folosirii limbajului in context didactic, deoarece permitere folosirea acestuia fara un proces initial de instalare sau configurare. Este o tendinta intre limbajele populare in prezent sa aiba abordari similarefootnoteUn Read Eval Print Loop (REPL) online pentru Scala: http://www.simplyscala.com/[34].

Contirbutii

Principalele contributii ale acestei teze sunt:

- O analiza amanuntita a principalelor alternative pentru dezvoltarea proiectelor AOP, Jason si JADE;
- Un model pentru executia concurenta in contextul limbajelor AOP, bazat pe modelul formal prezentat in [16]. Acest model este o descriere monadica a unitatilor de executie concurente, care permite o descriere naturala a compozitiei acestor unitati de executie;
- O implementare a modelului mentionat la punctul anterior peste platforma CLR, folosind sablonul *thread pool*. Aceasta abordare este in ton cu tendintele curente din comunitatea AOPL [3];

[2]Consultati http://altjs.org/ pentru o lista cuprinzatoare.
[3]Un REPL online pentru Haskell: http://tryhaskell.org/
[4]Un REPL online pentru F#: http://tryfsharp.org/

- Introducerea unui sistem de tipuri manifest pentu un AOPL, care permite dezvoltarea de proiecte solid utilizand tehnologiile orientate pe agenti. Si acest aspect este in ton cu tendintele curente din comunitatea AOPL: de curand au inceput sa fie publicate eforturi de dezvoltare a unor limbaje orientate pe agenti cu sistem de tipuri manifest [12];

- Introducerea *protocoalelor* pentru canalele de comunicare, care forteaza schimbul de mesaje prin intermediul canalelor de comunicatie sa respecte o schema definita de programator. Desi concepte similare exista si in alte limbaje, BLUEPRINT este primul limbaj cu aceasta caracteristica;

- Folosirea canalelor de comunicare asincrone si asimetrice, care au si rolul de puncte de sincronizare in programele agent. Acestea permit introducerea conceptului de *capabilitati* [9, 8], punand bazele explorarii securitatii in contextul AOPL;

- Sintaxa inspirata de limbajele din familia C, care face BLUEPRINT accesibil programatorilor incepatori;

- Compararea performantelor implementarii limbajului BLUEPRINT cu cea a implementarilor principalelor surse de inspiratie in dezvoltarea acestuia: Jason si JADE. Aceste doua solutii sunt cele mai populare alternative in lumea agentilor in prezent.

Publicatiile Autorului

[1] **Muscar, Alex**. Agent Oriented Programming: from Revolution to Evolution (**DBLP**). In *ICCSW*, pages 52–58, 2011.

[2] **Muscar, Alex**. Investigating F# as a development tool for distributed multi-agent systems. 752:32–36, 2011.

[3] **Muscar, Alex**. Towards a Pragmatic Approach of Agent Development, 6th South East European Doctoral Student Conference. In *6th South East European Doctoral Student Conference*, pages 320–326, 2011.

[4] **Muscar, Alex**. Agents for the 21st Century: the Blueprint Agent Programming Language. In *Online proceedings of the 1st International Workshop on Engineering Multi-Agent Systems*, pages 255–270, 2013.

[5] **Muscar, Alex**. Extending Jason with Promises for Concurrent Computation (**Springer**, **DBLP**). In Giancarlo Fortino, Costin Bădică, Michele Malgeri, and Rainer Unland, editors, *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 41–50. Springer Berlin Heidelberg, 2013.

[6] **Muscar, Alex**. Join Patterns for Concurrent Agents (position paper) (**ISI**). *SCPE journal*, 43(3):181–187, October 2013.

[7] **Muscar, Alex**. Agents in the Browser – Using Agent Oriented Programming for Client Side Development (**Springer**). In Amelia Badica, Bogdan Trawinski, and Ngoc Thanh Nguyen, editors, *Recent Developments in Computational Collective Intelligence*, volume 513 of *Studies in Computational Intelligence*, pages 79–90. Springer International Publishing, 2014.

[8] **Muscar, Alex** and Costin Bădică. A Functional Approach to Agent Development: Research Agenda (**IEEE**). In *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*, pages 380–385, 2011.

[9] **Muscar, Alex** and Costin Bădică. A Functional Take on Multi-Agent Systems Position Paper. *AIP Conference Proceedings*, 1389(1):865–868, 22 September 20 2011.

[10] **Muscar, Alex** and Costin Bădică. Investigating F# as a development tool for distributed multi-agent systems (*extended version*) (**IEEE**). *System Theory, Control, and Computing (ICSTCC)*, pages 1 – 6, 14-16 Oct. 2011 2011.

[11] **Muscar, Alex** and Costin Bădică. Exploring the Design Space of a Declarative Framework for Automated Negotiation: Initial Considerations (**Springer**, **DBLP**). In *AIAI (1)*, pages 264–273, 2012.

[12] **Muscar, Alex** and Costin Bădică. Towards a declarative framework for the specification of agent-driven auctions. *Engineering Intelligent Systems Journal*, 21(2/3), September 2013.

[13] **Muscar, Alex**, Laura Surcel, and Costin Bădică. Using Jason to Develop Declarative Prototypes of Automated Negotiations (**DBLP**). In *BCI (Local)*, pages 136–138, 2012.

Bibliography

[1] Costin Badica, Zoran Budimac, Hans-Dieter Burkhard, and Mirjana Ivanovic. Software agents: Languages, tools, platforms. *Comput. Sci. Inf. Syst.*, 8(2):255–298, 2011.

[2] Rafael H. Bordini, Jomi Fred Hübner, and Renata Vieira. Jason and the golden fleece of agent-oriented programming. In Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, editors, *Multi-Agent Programming*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 3–37. Springer, 2005.

[3] Rafael C. Cardoso, Jomi Fred Hübner, and Rafael H. Bordini. Benchmarking communication in actor- and agent-based languages. In Maria L. Gini, Onn Shehory, Takayuki Ito, and Catholijn M. Jonker, editors, *AAMAS*, pages 1267–1268. IFAAMAS, 2013.

[4] Cristiano Castelfranchi. Bye-bye agents? not. *IEEE Internet Computing*, 14:93–96, March 2010.

[5] Claudia V. Grigore and Rem W. Collier. Af-raf: an agent-oriented programming language with algebraic data types. In *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11*, SPLASH '11 Workshops, pages 195–200, New York, NY, USA, 2011. ACM.

[6] Carl Hewitt. Perfect disruption: The paradigm shift from mental agents to orgs. *IEEE Internet Computing*, 13:90–93, January 2009.

[7] Technical Committee International Organization of Securities Commissions. Regulatory issues raised by the impact of technological changes on market integrity and efficiency. Technical report, International Organization of Securities Commissions, 2011.

[8] Mark S. Miller, Ka-Ping Yee, and Jonathan Shapiro. Capability Myths Demolished. Technical report, Systems Research Laboratory, Johns Hopkins University, 2003.

[9] Mark Samuel Miller. *Robust composition: towards a unified approach to access control and concurrency control*. PhD thesis, Johns Hopkins University, Baltimore, MD, USA, May 2006. AAI3245526.

[10] Mattia Minotti, Giulio Piancastelli, and Alessandro Ricci. Agent-oriented programming for client-side concurrent web 2.0 applications. In José Cordeiro, Joaquim Filipe, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, and Clemens Szyperski, editors, *Web Information Systems and Technologies*, volume 45 of *Lecture Notes in Business Information Processing*, pages 17–29. Springer Berlin Heidelberg, 2010.

[11] Alessandro Ricci and Andrea Santi. Designing a general-purpose programming language based on agent-oriented abstractions: the simpal project. In *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11,*

AOOPES'11, NEAT'11, & VMIL'11, SPLASH '11 Workshops, pages 159–170, New York, NY, USA, 2011. ACM.

[12] Alessandro Ricci and Andrea Santi. Typing multi-agent programs in simpal. In Mehdi Dastani, F. Hübner, Jomi, and Brian Logan, editors, *Programming Multi-Agent Systems*, volume 7837 of *Lecture Notes in Computer Science*, pages 138–157. Springer Berlin Heidelberg, 2013.

[13] Andrea Santi. From objects to agents: Rebooting agent-oriented programming for software development. *Proceedings of The 12th European Agent Systems Summer School (EASSS 2010), 2010*, 2010.

[14] Andrea Santi, Marco Guidi, and Alessandro Ricci. Jaca-android: An agent-based platform for building smart mobile applications. In Mehdi Dastani, Amal El Fallah-Seghrouchni, Jomi Hübner, and João Leite, editors, *LADS*, volume 6822 of *Lecture Notes in Computer Science*, pages 95–114. Springer, 2010.

[15] Yoav Shoham. Agent-oriented programming. *Artif. Intell.*, 60:51–92, March 1993.

[16] Don Syme, Tomas Petricek, and Dmitry Lomov. The f# asynchronous programming model. In Ricardo Rocha and John Launchbury, editors, *PADL*, volume 6539 of *Lecture Notes in Computer Science*, pages 175–189. Springer, 2011.

[17] Gaku Yamamoto, Hideki Tai, and Hideyuki Mizuta. A platform for massive agent-based simulation and its evaluation. In Nadeem Jamali, Paul Scerri, and Toshiharu Sugawara, editors, *Massively Multi-Agent Technology*, volume 5043 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2008.