

Analiza și proiectarea aplicațiilor orientate-agent

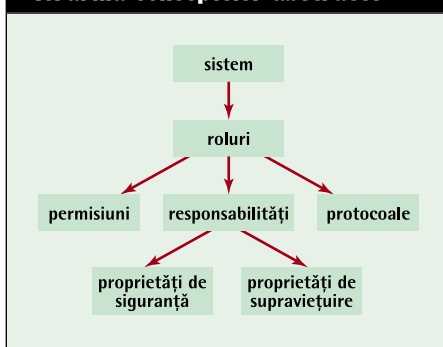
Metodologie și studiu de caz – Sabin-Corneliu Buraga

În prezentul articol ne vom ocupa de o metodologie utilă pentru analiză și proiectarea aplicațiilor orientate-agent, urmată de un studiu de caz al unui agent utilizat pe Web în domeniul comerțului electronic.

În ultimii douăzeci-treizeci de ani ingineria software s-a concentrat asupra dezvoltării unor abstractizări utile modelării și dezvoltării de sisteme complexe. Ca exemple de astfel de abstractizări se pot menționa abstractizările procedurale, tipurile de date abstracte și, mai recent, obiectele. Agenții reprezintă un alt mod avansat de abstractizare, dar pentru dezvoltarea de sisteme orientate-agent nu se pot adopta tehnicile actuale.

În aplicarea oricărei metodologii, analistul trece de la abstract la concret. Metodologia-agent propusă de M.Wooldridge și descrisă mai jos încurajează dezvoltatorii să considere conceperea unui sistem bazat pe agenți ca un proces de *proiectare organizațională*. Conceptele principale pot fi divizate în două categorii: *abstracte* și *concrete*. Entitățile abstracte sunt necesare în timpul activității de analiză, iar cele concrete sunt utilizate mai ales în procesul de proiectare și sunt în strânsă legătură cu faza de implementare a aplicației. Vom prezenta în continuare atât faza de analiză, cât și cea de proiectare a aplicațiilor orientate-agent.

Figura 1. Ierarhia conceptelor abstracte



Concepte abstracte.

Cea mai abstractă entitate în ierarhia conceptelor este *sistemul*. În cadrul acestei metodologii sistemul poate fi înrudit cu termenii „societate” sau „organizație”. Astfel, putem gândi sistemul ca o societate/organizație artificială.

Ideea de sistem văzut ca o societate este utilă atunci când ne concentrăm asupra următorului nivel din ierarhia conceptuală (vezi figura de mai sus): nivelul *rolurilor*. Fiecare agent poate juca un anumit rol în cadrul sistemului, desigur putând exista agenți multipli care să joace același rol. Un rol este definit de trei atribute: *responsabilități*, *permisiuni* și *protocoale*. Responsabilitățile determină funcționalitatea agentului, putând fi divizate în *proprietăți de supraviețuire* (descriu stările de evoluție a agentului în funcție de condițiile de mediu) și *proprietăți de siguranță* (invariante, descriu starea globală a sistemului sau diverse restricții impuse). Pentru realizarea responsabilităților, un rol are, uzual, asociată o mulțime de permisiuni (drepturi) identificând resursele disponibile unui rol, astfel permisiunile tind a fi *resurse de informații*. Un rol de asemenea poate genera informații. Interacțiunea unui rol cu altele se realizează prin intermediul protocolurilor.

Analiza

Obiectivul fazei de analiză este înțelegerea optimă a sistemului și a structurii sale, fără a se face nici o referire la detaliile de implementare. În acest caz, procesul de înțelegere este capturat de modul de *organizare* a sistemului. Putem vedea organizația ca o colecție de roluri având diverse relații între ele. Pentru a defini organizația este suficient să definim rolurile sale și interacțiunea lor. Astfel, putem diviza analiza în: *modelul rolurilor* și *modelul interacțiunilor*.

Modelul rolurilor identifică rolurile principale (cheie) din sistem, un rol fiind privit ca o descriere abstractă a unei funcții a unei entități. Rolurile au două tipuri de atribute: permisiunile (drepturile) asociate unui rol și res-

ponsabilitățile unui rol. Permișiunile vor identifica resursele care pot fi folosite și limitele de funcționare în cadrul unui rol. Resursele se referă doar la informațiile sau cunoștințele deținute de un agent. Notația formală pentru exprimarea permisiunilor va fi cea bazată pe *FUSION* (vezi mai jos).

Funcționalitatea unui rol este definită de responsabilitățile sale, acestea putând fi împărțite în două categorii: de supraviețuire și de siguranță. Cele de supraviețuire sunt necesare mai ales la modelarea proprietăților sistemelor reactive.

Ilustrăm conceptele asociate rolurilor printr-un exemplu simplu: fie rolul de filtru de cafea, scopul acestuia fiind să-și păstreze întotdeauna plin cu cafea recipientul. Responsabilitățile de supraviețuire ale rolului pot fi:

- de câte ori nu mai este cafea, umple recipientul;
- de câte ori s-a făcut cafea proaspătă, anunță doritorii.

Aceste responsabilități vor fi specificate printr-o *expresie de supraviețuire* care definește ciclul de viață a rolului. Expresiile sunt de fapt expresii regulate la care se adaugă operatorul ω . Operatorii permiși sunt cei din tabelul de mai jos.

Operatorii folosiți în metodologia orientată-agent

Operator	Descriere
$x.y$	x urmat de y
x^*	x apare de 0 sau mai multe ori
x^+	x apare măcar o dată
x^ω	x apare de o infinitate de ori
$[x]$	x este opțional
$x \parallel y$	x și y decurg în paralel

Expresiile de supraviețuire definesc traiectoriile posibile de execuție prin intermediul activităților și interacțiunilor asociate unui rol, forma lor generală fiind:

$\text{Numero1} = \text{expresie}$

Componentele atomice ale unei expresii sunt numite *protocele*.

Responsabilitățile rolului filtrului de cafea vor putea fi scrise astfel:

FCafea =
(Umple.Informează.Verifică.Așteaptă)?

Sunt executate la infinit protocelele *Umple*, *Informează*, *Verifică* și *Așteaptă*, în această ordine.

În multe cazuri, vom avea nevoie și de anumiți *invarianți* care să fie menținuți pe timpul execuției unui rol. Acești invarianți vor fi numiți *condiții de siguranță*. În cazul nostru, expresia de siguranță asociată va fi: $\text{cantitCafea} \geq 0$

Responsabilitățile se vor aplica pentru toate stările execuției sistemului.

Schema unui rol: FCafea

Descriere: rolul asigură ca există suficientă cafea, informând când a fost făcută cafea proaspătă

Protocele: Umple, Informează, Verifică, Așteaptă

Permișiuni: citește FiltruCafea, stareCafea modifică cantitCafea

Responsabilități

De supraviețuire: FCafea = (Umple.Informează.Verifică.Așteaptă)?

De siguranță: $\text{cantitCafea} \geq 0$

Modelul rolurilor va fi specificat printr-un set de *scheme de roluri*, după următorul șablon:

Schema unui rol: nume rol

Descriere:

Protocele:

Permișiuni:

Responsabilitati

De supravietuire:

De siguranta

Pentru exemplul nostru vom avea schemă din caseta alăturată.

Modelul interacțiunilor. Legăturile dintre diversele roluri vor fi reprezentate de modelul interacțiunilor, constând dintr-un set de *definiții de protocele*, unul pentru fiecare tip de interacțiune. Aceste definiții au următoarele atribute:

- *scop*: descriere scurtă a naturii interacțiunii (de exemplu: „cerere de informații” ori „activitate de planificare”);
- *inițiator*: rolul sau rolurile care vor iniția interacțiunea;
- *destinatar*: rolul sau rolurile cu care inițiatorul interacționează;
- *intrări*: informațiile utilizate de inițiator în cadrul protocolului;

- *ieșiri*: informațiile furnizate de destinatarul protocolului în cadrul interacțiunii;
- *procesare*: scurtă descriere a oricărei procesări pe care o face inițiatorul în cursul interacțiunii.

Procesul de analiză. Faza de analiză în metodologia orientată-agent poate fi sumarizată de următorii pași:

- (1) identificarea rolurilor din sistem rezultând un model prototip al rolurilor;
- (2) pentru fiecare rol, identificarea și documentarea protocelelor asociate, rezultând un model de interacțiune;
- (3) utilizând modelul interacțiunilor se elaborează modelul rolurilor, rezultând un model elaborat complet;
- (4) iterarea pașilor 1.-3.

Proiectarea

Procesul clasic de proiectare constă din transformarea modelelor abstracte obținute în faza de analiză în modele cu un nivel redus de abstractizare astfel încât să fie ușor implementabile. Procesul de analiză și proiectare orientat-agent are în vedere modul de cooperare a societății de agenți pentru realizarea scopurilor la nivelul sistemului și modelarea fiecărui agent individual. Despre cum își realizează agenții serviciile se va răspunde în faza de implementare, depinzând de domeniul de aplicabilitate a sistemului.

Activitatea de proiectare presupune imaginarea a trei modele. *Modelul agent* identifică tipurile de agenți care vor exista în sistem și instanțele agenților. *Modelul serviciilor* specifică principalele servicii asociate fiecărui tip de agenți. Ultimul model este cel al *cunoștințelor* asociate fiecărui tip.

Modelul agent. Poate exista o corespondență 1-la-1 între roluri (identificate de modelul

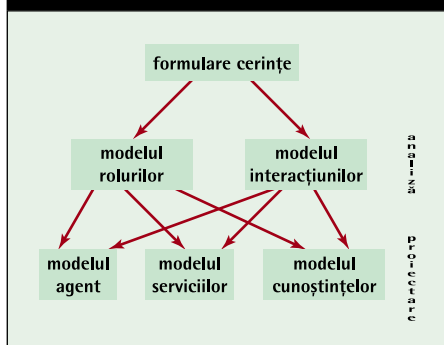
rolurilor) și tipurile de agenți, însă în realitate proiectantul poate alege să restrângă numărul tipurilor de agenți, în funcție de similaritățile rolurilor sau din considerente de eficiență. Nu trebuie uitată nici coerența tipurilor de agenți (cât de ușor poate fi înțeleasă funcționalitatea agenților).

Modelul agent este definit utilizând un *arbore de tipuri de agenți*, fiecare nod rădăcină corespunzând rolurilor, iar celelalte noduri corespunzând tipurilor de agenți. Tipurile de agenți pot fi *adnotate*. O adnotare n înseamnă că vor exista exact „ n ” agenți de acel tip în cadrul implementării sistemului. O adnotare „ $m..n$ ” înseamnă că pot exista minim m instanțe și maxim n instanțe ale celui tip de agent ($m < n$). O adnotare „*” specifică faptul că vor fi zero sau mai multe instanțe, iar „+” că va exista cel puțin un agent dintr-un anumit tip.

Conceptul de moștenire nu există în cadrul acestui model, putând fi regăsit la faza de implementare a sistemului.

Modelul serviciilor. Acest model identifică serviciile asociate fiecărui rol și proprietățile lor. Printr-un serviciu se înțelege o *funcție* a unui agent, în termenii metodologiei orientate-obiect corespunzând unei metode. Serviciile însă nu vor fi disponibile pentru alți agenți, ci vor forma un bloc unic de activități pe care le va executa un agent. Proprietățile unui serviciu sunt: *intrările*, *ieșirile*, *pre-condițiile* și *post-condițiile*. Intrările și ieșirile vor fi derivate din modelul protocelelor. Pre- și post-condițiile vor reprezenta constrângeri impuse serviciilor, derivate din proprietățile de siguranță ale unui rol. Prin definiție, fiecare rol va avea asociat cel mult un serviciu.

Figura 2. Relațiile dintre modelele metodologiei orientate-agent



Serviciile pe care le poate oferi un agent vor deriva din lista de protocoale și responsabilități asociate unui rol. Proiectantul este liber să realizeze serviciile în orice cadru de implementare dorit (de exemplu, serviciile pot fi metode într-un limbaj orientat-obiect sau pot fi descompuse într-un anumit număr de metode).

Modelul cunoștințelor. Modelul definește legăturile de comunicare care vor exista între tipurile de agenți, fiind reprezentate prin căi de comunicare și nu prin formatul sau conținutul mesajelor care pot fi vehiculate. Scopul modelului este să identifice orice problemă de comunicare. Pe bazele modelului cunoștințelor, faza de analiză se poate relua pentru a se înlătura problemele de comunicare.

Modelul va construi un graf orientat, având ca noduri tipuri de agenți și ca arce legături de comunicare. Dacă există arc de la *a* la *b*, atunci *a* trimite mesaje lui *b*. Modelul va ține seama de modelele precedente (modelul rolurilor, protocoalelor și agenților).

Procesul de proiectare. Faza de proiectare a metodologiei orientate-agent are următorii pași:

- (1) crearea unui model agent: se agregă rolurile ca tipuri de agenți și se formează o ierarhie de tipuri, adnotându-se apoi instanțele fiecărui tip de agent;
- (2) dezvoltarea unui model al serviciilor, examinându-se protocoalele și proprietățile rolurilor;

(3) crearea unui model al cunoștințelor pornind de la modelul interacțiunilor și modelul agent.

Relațiile dintre modelele metodologiei propuse pot fi sintetizate de figura 2.

Concluzii

Există, desigur, și alte propuneri de metodologii pentru analiza și proiectarea aplicațiilor folosind tehnologii orientate-agent. O parte ia în considerație tehnicile existente de modelare obiectuală pentru a fi adoptate și în cazul agenților, alte direcții vizează crearea unor limbaje de modelare compozițională care să verifice la nivel formal structura și funcționarea sistemelor compuse din agenți.

Sistemele complexe multi-agent bazate pe convingeri, dorințe și intenții vor trebui modelate prin alte tehnici (e.g. *DMARS - Distributed Multi-Agent Reasoning System*).

Metodologia orientată-agent a avut ca punct de start modelele orientate-obiect tradiționale, adaptate și extinse pentru a îndeplini cerințele impuse de noua abordare.

Studiu de caz: Un agent Web pentru comerțul electronic

Paradoxal, pe cât de ușor poate fi folosit Web-ul de către oameni, pe atât de greu informațiile sale pot fi manipulate de agenți (marcajele HTML oferă posibilități avansate de afișare a paginilor Web, dar nu dau prea multe indicii despre conținutul acestora). Proiectanții unui agent Web inteligent trebuie să ia în considerare următoarele aspecte:

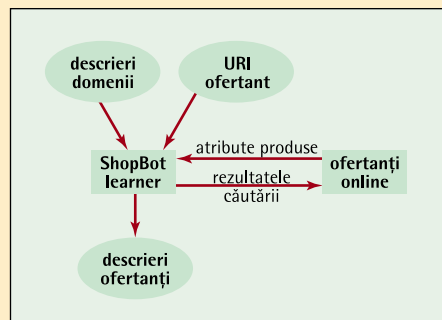
- Cât de mult poate înțelege un agent informația disponibilă pe Web?
 - Cunoștințele agentului sunt suficiente pentru a oferi diverse servicii utilizatorilor?
 - Agenții Web pot în mod automat extrage informații de pe orice server Web sau este necesar un marcaj special?
 - Care aspecte ale Web-ului facilitează utilizarea agenților?
- Vom prezenta în cele ce urmează un exemplu de agent Web, independent de domeniu, care învață ce să cumpere de la o serie de ofertanți de situri de comerț electronic. După ce urmează un proces de învățare, este capabil de a vizita diverși ofertanți de produse software și CD-uri multimedia, extrăgând informațiile despre aceștia și oferind utilizatorului un sumar al lor. Acest agent se numește *ShopBot*, a fost dezvoltat la Universitatea din Washington și posedă capabilități de procesare a limbajului natural, necesitând cunoștințe minime despre diverse tipuri de produse.

Sarcinile agentului. Agentul Web va trebui să ofere utilizatorilor asistență în cumpărarea de produse disponibile pe Internet, sarcinile lui principale fiind:

- ajutor acordat utilizatorilor în decizia cumpărării unor produse;
 - găsirea specificațiilor și prezentărilor acestor produse;
 - recomandarea de produse pe baza unor caracteristici (calitate, fiabilitate, preț etc.);
 - compararea prețurilor pentru a găsi cel mai bun preț pentru produsul dorit;
 - monitorizarea listelor de tip „What's new” sau a altor surse pentru descoperirea unor noi informații relevante;
 - monitorizarea apariției ofertelor speciale sau a ieftinirilor.
- Agentul *ShopBot* va putea fi folosit mai ales pentru compararea

produselor și înșirarea utilizatorului asupra celui mai bun produs.

Prezentare generală. *ShopBot* operează în două faze: o *fază de învățare*, în care un modul denumit *learner* creează o descriere a furnizorilor și a tipurilor de produse și o *fază de comparare a produselor*, în care un modul *shopper* utilizează descrierile generate în prima fază pentru a ajuta o persoană în decizia de a realiza cumpărături având cel mai bun (mic) preț. Faza de învățare analizează serverele Web ale ofertanților de produse pentru a învăța o descriere simbolică a fiecăruia, algoritmul de învățare fiind dat succint mai jos:



Faza de creare și învățare a descrierilor ofertanților

Date de intrare:

un model incomplet de domeniu:

exemple de produse: P_1, P_2, \dots, P_n

attribute ale produselor: manufacturer, name, type etc.

(exemplu: $manufacturer(P_1) = \text{“Microsoft”}$, $name(P_1) = \text{“Encarta”}$, $type(P_1) = \text{“multimedia enciclopedia”}$)

URI-ul paginii Web a ofertantului

Algoritmul:

Se accesează pagina Web dată pentru detectarea produselor dorite și se furnizează un set de descrieri simbolice ale ofertanților acelor produse (e.g. MS ENCARTA 2001 - CD-ROM... `EDU 1033... ` \$99.95...).

Principala separație între modelele deja concepute este cea dată de clasificarea modelelor ca *externe* sau *interne*. Modelele externe prezintă o vedere de ansamblu, la nivelul sistemului, principalele componente vizibile fiind agenții și relațiile și interacțiunile dintre ei, incluzând relațiile de moștenire și agregare, utile în abstractizarea structurii agenților. Modelele interne, prin contrast, iau în considerație atributele agenților: convingerile, scopurile, planurile lor.

Ca exemple de modele externe pot fi date *modelul agent* (descrie clasele și instanțele agenților) și *modelul interacțiunilor*. Modelele interne sunt extensii directe ale modelelor obiect și dinamic din cadrul metodologiei orientate-obiect.

Cercetările continuă pentru a concepe o metodologie clară, concisă și flexibilă pentru analiza și proiectarea aplicațiilor orientate-agent.

Sabin-Corneliu Buraga este doctorand în Computer Science la Universitatea „A.I. Cuza” din Iași, putând fi contactat prin e-mail la adresa busaco@infoiasi.ro si pe Web la http://www.infoiasi.ro/~busaco. ■ 76

Referințe bibliografice

- D.Amor - „The E-Business (R)evolution”, Prentice-Hall, 1999
 G.Booch - „Object-Oriented Analysis and Design”, Addison-Wesley, 1994
 J.Bradshaw - „Software Agents”, AAAI Press, 1997
 S.Green, F.Somers - „Software Agents: A Review”:
http://www.cs.tcd.ie/research_groups/aig/iag/iag.html
 K.Höök et al. - „Edited Adaptive Hypermedia: Combining Human and Machine Intelligence to Achieve Filtered Information”, Flexible Hypertext Workshop, Hypertext'97 Proceedings, ACM, 1997
 C.Petrie - „Agent-Based Engineering, the Web and Intelligence”, IEEE Expert, dec.1996
 A.Rao et al. - „Formal Models and Decision Procedures for Multi-Agent Systems”, Technical Report 61, Australian Artificial Intelligence Institute, 1995
 D.Weerasooriya et al. - „Design of a Concurrent Agent-Oriented Language”, in „Intelligent Agents: Theories, Architectures and Languages”, LNAI890, Springer Verlag, Amsterdam, 1995
 M.Wooldridge, N.Jennings - „Intelligent Agents: Theory and Practice”, Knowledge Engineering Review Journal, 10(2), 1995
 M.Wooldridge, N.Jennings - „Pitfalls of Agent-Oriented Development”, Agents'98 Proceedings, Minneapolis, 1998
 M.Wooldridge, N.Jennings, D.Kinny - „A Methodology for Agent-Oriented Analysis and Design”, Agents'98 Proceedings, Minneapolis, 1998
 *** - „AgentWeb”: <http://www.cs.umbc.edu/agents>
 *** - „ShopBot”: <http://www.cs.washington.edu/research/shopbot>

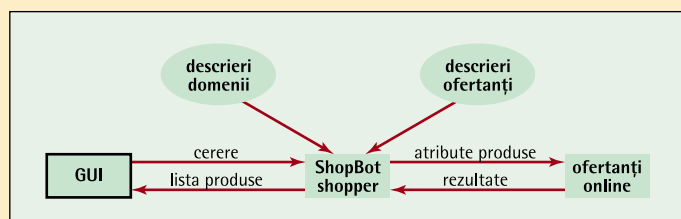
Procedura de învățare a agentului

Descrierile ofertanților au forma următoare:

- URI-ul paginii conținând formularul unui index de căutare;
- funcție de asociere a atributelor produsului la câmpurile formularului electronic;
- funcții de extragere a datelor despre produse din paginile returnate de index;
- funcție de recunoaștere a paginilor de eroare (e.g. „Product not found.”);
- funcție de ignorare a antetelor și a altor date adiționale neimportante (e.g. designerul paginii);
- funcție de extragere a descrierilor de produse din textul rămas.

În faza a doua se folosește o procedură de extragere a produselor pentru o varietate de ofertanți și se prezintă utilizatorului o sumarizare a rezultatelor obținute

Proprietăți ale mediului web



Faza de asistare a cumpărării de produse de pe Web

Conținutul paginilor WWW ale unor ofertanți *online* se bucură de următoarele proprietăți care pot fi exploatate de un agent web:

- *navigarea* - Magazinele Web sunt proiectate în așa fel încât consumatorii să găsească rapid produsele (asigurând mecanisme de navigare ușoară, indecși de căutare, harta sitului etc.).
- *uniformitatea* - Ofertanții utilizează un mod uniform și consistent de prezentare a tuturor produselor disponibile.

- *separarea verticală* - Se utilizează spații albe sau linii de demarcație între diverse (tipuri de) produse pentru a putea fi ușor reperate de clienți (de exemplu, jucăriile sunt separate de produsele electronice sau de CD-uri).

ShopBot preferă siturile Web care au indecși de căutare automată a unor produse pentru ușurința exploatarei, ignorând furnizorii care nu oferă aceste facilități. ShopBot se bazează pe marcajele HTML și va eșua în evaluarea formularelor grafice sau bazate pe applet-uri Java.

Comparații cu alți agenți

Principala deosebire față de alți agenți similari este că ShopBot nu folosește marcaje speciale pentru detectarea produselor ofertanților de pe Web. ShopBot a fost inspirat dintr-o serie de agenți manipulând informații structurale cum ar fi *SoftBot* care extrage diverse date din rezultatele unor comenzi UNIX precum `ls` sau `find`. Desigur, există agenți pentru analiza nestructurată a paginilor Web, utilizați mai ales în contextul navigării asistate sau a căutărilor euristice (cazul roboților Web).

Aproape toți agenții de învățare învață despre interesele utilizatorului și mai puțin despre resursele externe la care au acces. Excepția cea mai exemplificatoare este *ILA* (Internet Learning Agent) care învață să „înțeleagă” sursele externe de informație prin catalogarea lor pe baza unor criterii predefinite. ShopBot a preluat de la ILA acest aspect.

Un alt agent destinat asistării comerțului electronic este *Bargain-Finder*, dependent însă de tipul și de structura magazinului virtual al ofertantului.

Desigur, se așteaptă ca în viitorul apropiat să fie dezvoltate și alte tipuri de agenți facilitând cumpărăturile pe Web, prin extragerea informațiilor multimedia despre produsele disponibile, apelând, de exemplu, la rețele neurale sau algoritmi genetici.