Modelul arhitectural blackboard

Manaţe Bogdan

03/27/2011

1 Introducere

Dezvoltarea sistemelor multi-agent mari este o sarcină complexă care implică mai multe procese, cum ar fi realizarea cerinţelor, arhitectura, designul şi implementarea acestor sisteme. În particular, designul arhitecturii este critic pentru a face faţă dimensiunii şi complexităţii în continuă creştere a acestor sisteme. Sistemele multi-agent care accesează un depozit central de date sunt de obicei bazate pe şablonul arhitectural blackboard.

Sistemele software construite ca un grup de agenţi inteligenţi şi autonomi promit să ofere putere de calcul pentru a rezolva probleme dificile. Cu toate acestea, sistemele multi-agent complexe sunt dificil de realizat. În unele sisteme agenţii pot rezolva probleme intensive, dar pentru acest lucru toţi agenţii trebuie să fie cunoscuţi de la începutul execuţiei programului. În alte sisteme, agenţii nu se cunosc de la început, iar pentru a putea interacţiona, aceştia trebuie să se înregistreze la un serviciu de descoperire. O dată ce un agent are cunoştiinţă existenţa altor agenţi în acelaşi mediu cu el, el poate să folosească serviciile sau cunoştiinţele dobândite de acei agenţi. În sistemele de acest gen, un agent îşi poate asuma un rol de control, pentru a administra ceilalţi agenţi prezenţi în sistem.

În dezvoltarea sistemelor multi-agent mari, trebuie să se ţină cont şi de caracteristicile sistemului, cum ar fi performanţa, reutilizabilitatea, adaptibilitatea, securitatea şi mobilitatea. Aceste proprietăţi care au nevoie de strategii pentru controlul logicii şi datelor, trebuie să fie luate în considerare la începutul proiectării arhitecturii. În general, folosirea şabloanelor software arhitecturale poate facilita dezvoltarea sistemelor multi-agent complexe.

Invocarea implicită este un şablon arhitectural des întâlnit, care are ca avantaj decuplarea componentelor, care pot fi în acest caz obiecte care reprezintă agenţi. Acest şablon este potrivit pentru aplicaţile care folosesc colecţii de componente slab cuplate, fiecare dintre componente execută anumite operaţii sau fac parte dintr-un proces amplu care are ca rezultat o operaţiune.

Obiectele şi agenţii au multe caracteristici comune fiind de asemenea diferite. Din perspectiva tradiţională orientată obiect, un obiect nu este autonom. Comportamentul său interior putând fi modificat din exterior. Un obiect nu este capabil să decidă ce trebuie să facă într-o anumită situaţie. În aplicaţiile

tradiţionale dezvoltate pe principiul OOP, mediul nu este specificat explicit şi poate fi incapsulat ca atribute interne. Din perspectiva sistemelor multi-agent poate avea o stare activă şi poate integra fire de execuţie permiţându-i să execute diverse sarcini computaţionale.

2 Arhitectura blackboard bazată pe evenimente

Arhitectura blackboard este contruită prin integrarea a două şabloane arhitecturale: şablonul blackboard şi şablonul invocării implicite. În această arhitectură metodele de control sunt separate de agenţii cu rol de control pentru ca aceste metode să se poată schimba independent.

Modelul arhitectural blackboard este utilizat pentru aplicaţiile care trebuie să gestioneze probleme non-deterministe. Pentru majoritatea acestor probleme nu există o soluţie bazată pe un anume algoritm, iar o soluţie aproximativă este acceptată. Acest model arhitectural este folosit în cadrul diferitelor sisteme software folosite în majoritatea domenilor. Acest model este format din trei componente: componenta de control, blackboard-ul şi surse de cunoştinţe.

Sursele de cunoştinţe pot fi alcătuite din agenţi specializaţi în rezolvarea anumitor probleme. Aceştia procează cunoştinţele de care au nevoie şi le fac publice componentei blackboard fără a avea nevoie de interacţiunea altor agenţi. Sursele de cunoştinţe sunt alcătuite din două subcomponente majore: condiţii şi acţiuni. Condiţile sunt folosite pentru a determina când agentul poate contribui cu ceva. Când condiţile sunt îndeplinite se invocă acţiunile specifice. Acţiunile includ modificarea sau plasarea noilor date în cadrul componentei blackboard.

Componenta blackboard este sursa tuturor datelor asupra cărora va opera o sursă de cunoştinţe fiind de asemenea şi destinaţie pentru datele generate de sursele de cunoştinţe.

Componenta de control este reprezentată de un manager care administrează accesul surselor de cunoştinţe la componenta blackboard. Rolul componentei de control poate fi luat de un agent care este specializat în administrarea şi planificarea resurselor. Controlul se poate realiza prin invocarea directă a surselor de cunoştinţe sau prin apelarea de la distanţă a metodelor. Schimbarea dinamică a controlului presupune ca toţi agenţii care substituie componenta de control să transmită succesorilor metodele de control, pentru a evita rezultatele eronate datorate schimbării de control.

Sistemele complexe multi-agent se doresc a fi stabile şi să aibă o durată de viaţă considerabilă. Componenta de control este o parte critică a sistemului. În cazul în care unul dintr agenţi generează o eroare în timpul rulării, sistemul trebuie să ruleze în continuare. Rolul de componentei trebuie să poată fi luat de agenţii specializaţi fără a genera erori în sistem.

După definirea sistemului, trebuie avută în vedere şi mentenanţa sistemului. Arhitectura sistemul trebuie să suporte schimbările intervenite la nivelul agenţilor. De asemenea, arhitectura sistemelor multi-agent, trebuie să facă faţă evoluţiei astfel încât impactul asupra componentelor sistemului să fie minim. Noi agenţi trebuie să poată fi adăugaţi în sistem ca surse de cunoştinţe sau

componente de control, iar agenţii deja existenţi în sistem trebuie să poată să-şi încheie activitatea fără să afecteze funcţionalitatea sistemului.

2.1 Invocarea implicită

Şablonul arhitectural care se bazează pe invocarea implicită este potrivit pentru aplicaţii care sunt alcătuite din componente care se doresc a fi slab cuplate. În cadrul acestui şablon fiecare agent se înregistrează la anumite evenimente care sunt declanşate de alţi agenţi. Când evenimentul se declanşează, toţi agenţii care sunt înregistraţi la acest eveniment vor fi notificaţi.

În cadrul invocării implicite, agentul care anunţă evenimentul nu cunoaşte agenţii înregistraţi la acest eveniment. De asemenea agentul nu cunoaşte nici ordinea procesării evenimentului. Invocarea implicită are mai multe avantaje:

- agenţii sunt independenţi unul faţă de celălalt.
- interacţiunea este separată.
- nu există referinţe directe la ceilalţi agenţi.
- un nou agent se poate introduce în sistem prin înregistrarea la evenimentele dorite.
- se pot introduce noi tipuri de agenţi fără a modifica restul agenţilor.

De asemenea invocarea implicită prezintă şi unele dezavantaje:

- nu există control asupra secvenţei de declanşare a evenimentelor.
- necesitatea unei componente centrale de control care trebuie să administreze evenimentele şi agenţii înregistraţi la evenimente.
- folosirea evenimentelor poate genera erori dacă folosite fire de execuţie.

2.2 Compunerea

Şabloanele arhitecturale sunt folosite pentru rezolvarea celor mai frecvente probleme. În cadrul arhitecturii unui sistem este posibil să se folosească mai multe şabloane pentru rezolvarea cerinţelor sistemului. Compunerea mai multor şabloane arhitecturale are ca rezultat un nou şablon arhitectural. Din compunerea şablonului blackboard şi a şablonului invocării implicite rezultă şablonul arhitectural blackboard bazat pe evenimente. Figura 1 prezintă diagrama arhitecturii blackboard bazată pe evenimente. Datele conţinute de blackboard sunt alcătuite din baza de cunoştinţe şi protocolalele de control. Protocoalele de control pot fi accesate numai de agenţii care au rol de control, pe când baza de cunoştinţe poate fi accesată şi de agenţii care joacă rolul de surselor de cunoştinţe.

Dacă se consideră un sistem online de licitaţii cu agenţi înregistraţi ca licitatori şi vânzători. Licitatorii pot intra sau ieşi din sistem când doresc, iar

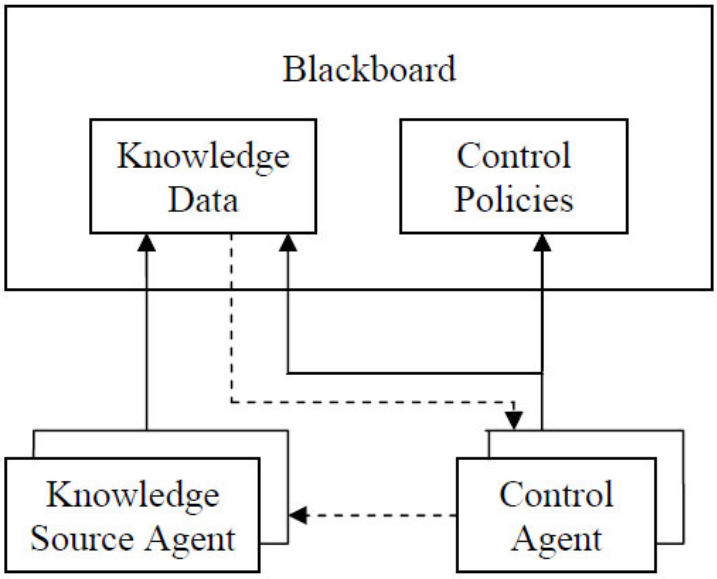

Figure 1: Diagrama arhitecturi blackboard bazată pe evenimente

vănzătorii pot fi înlocuiţi de către alţi vânzători. În acest exemplu blackboard-ul conţine informaţii despre produsele care sunt licitate şi regulile licitaţiei. Vânzătorul este reprezentat de un agent cu rol de control, acesta putând fi înlocuit în cazul unei erori fără a avea consecinţe asupra funcţionării sistemului. Cumpărătorii sunt reprezentaţi de agenţi cu rol de surse de cunoştinţe, aceştia se pot înregistra pentru a vizualiza anumite produse şi pentru a urmării desfăşurarea licitaţiei pentru produsele care prezintă interes.

Invocarea implicită este potrivită pentru aplicaţii care folosesc un număr mare de agenţi şi pentru care se doreşte decuplarea acestora. Prin incorporarea acestui şablon arhitectural cu şablonul blackboard, agenţii sunt decuplaţi, comunicarea realizându-se prin evenimente. Agentul vânzător şi agentul cumpărător nu se cunosc, dar agentul vânzător poate controla licitaţia prin evenimente.

3 Consecinţe

Implementarea sistemelor multi-agent folosind arhitectura blackboard bazată pe evenimente promovează decuplarea componentelor. Metodele de control sunt separate atât de agentul cu rol de control cât şi de datele şi logica sistemului. Această separare este importantă deoarece permite metodelor de control să fie independente de agentul cu rol de control. Combinarea lor ar face sistemul mai complex şi mai dificil de implementat.

Fiecare agent trebuie să se înregistreze în sistem ca o sursă de cunoştinţe. Înregistrarea îi permite sistemului să verifice identitatea agentului şi să-i permită

accesul doar la anumite părţi din sistem.

Mai mulţi agenţi cu rol de control pot să ruleze în paralel, acest lucru ducând la eliminarea blocajelor şi îmbunătăţeşte stabilitatea sistemului. Arhitectura blackboard bazată pe evenimente permite configurarea dinamică a agenţilor din sistem. Atât agenţii cu rol de control, cât şi agenţii cu rol de surse de cunoştinţe pot fi adăugaţi sau eliminaţi din sistem, iar datorită faptului că aceştia nu au referinţă directă unul către celălalt sistemul este reutilizabil şi uşor de menţinut.

Această arhitectură prezintă şi unele dezavantaje, cum ar fi probleme de performanţă introduse de manipularea evenimentelor şi dificultatea de depanare. Din cauza nondeterminismului şi mediului concurent controlul procesării evenimentelor este greu de controlat.

References

[1] Yariv Aridor and Danny B. Lange. Agent Design Patterns: Elements of Agent Application Design. Proceedings of the International Conference on Autonomous Agents, ACM Press, 1998.

[2] Paulo Alencar, Donald Cowan, Jing Dong, and Carlos Lucena, A Pattern-Based Approach to Structural Design Composition. Proceedings of the IEEE 23rd Annual International Computer Software & Applications Conference (COMPSAC), pp160- 165, Phoenix, USA, October 1999.

[3] G. Booch, J. Rumbaugh, and I. Jacobson. The Unified Modeling Language User Guide. Addison- Wesley, 1999.

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal. Pattern-Oriented Software Architecture: A System of Patterns, John Wiley & Sons, 1996.

[5] Daniel D. Corkill. Blackboard Systems. AI Expert. 6(9):40-47. September 1991.