Universitatea de Vest Timisoara,
Facultatea de Matematică-Informatică
Specializarea: Inginerie Software

# Lucrare de dizertație

(Raport intermediar)

Coordonator ştiințific: Prof. Dr. Viorel Negru
Assist. Drd. Ovidiu Aritoni

Autor: Manațe Bogdan
2011

Universitatea de Vest Timisoara,
Facultatea de Matematică-Informatică
Specializarea: Inginerie Software

# Model multi-agent pentru sisteme senzitive la context

Coordonator ştiinţific: Prof. Dr. Viorel Negru
Assist. Drd. Ovidiu Aritoni

Autor: Manațe Bogdan
2011

# Conținut

# 1. Introducere

Inteligența ambientală implică sintetizarea datelor dintr-o largă gamă de surse pentru a prezenta un comportament adaptiv relevant, fără implicarea directă a utilizatorului. Sintetizarea se face pe date provenite de la dispozitive de masură independente sau alte surse de date. În zilele noastre sistemele inteligente sunt omniprezente în toate domeniile, fiind prezente chiar şi în viața noastră de zi cu zi. Mediile inteligente se pot integra peste tot, de la case de locuit prevăzute cu dispozitive automate de control, la birouri sau hoteluri care au din ce în ce mai multe dispozitive inteligente pentru a asista muncitorii în activitățiile pe care le desfăşoară, pentru a economisii energie sau pentru a satisface preferințele clienților.

Proiectarea şi dezvoltarea de software pentru sistemele abientale inteligente implică provocări dificile în funcție de diversitatea dispozitivelor şi canalele de comunicație. Deoarece sistemele ambientale au cerințe imprevizibile şi sunt senzitive la context, sistemele software trebuie să suporte schimbări dinamice. Caracteristicile unui sistem ambiental multi-agent orientat pe aspecte sunt prezentate ca un mecanism de raționament pentru comportamente interdependente.

Sistemele ambientale inteligente vizionare se bazează pe maşini de calcul prevăzute cu un număr mare de procesoare şi senzori de mici dimensiuni integrați în obiectele utilizate în viața de zi cu zi, acest lucru ducând la dispariția dispozitivelor tradiționale şi a modului de prelucrare a informației. În cadrul sistemelor ambientale inteligente există un grad ridicat de partajare a datelor între unități larg distribuite, structurate pentru a promova colaborarea, auto-sincronizarea, răspunsuri rapide la informații noi, adaptabilitate şi sustenabilitate.

Proiectarea sistemelor ambientale inteligente reprezintă o provocare pentru arhitecții software. Sistemele de acest tip implică participarea dispozitivelor heterogene, care formează un sistem deschis, dinamic în cadrul căruia resursele disponibile, contextul şi activitățiile se schimbă continuu. Sistemele multi-agent prezintă o abordare naturală şi puternică pentru a proiecta sisteme abientale inteligente care să funcționeze în medii complexe.

Cele mai importante caracteristicii ale agenților inteligenți sunt anticiparea evenimentelor şi adaptarea la schimbările mediului în care aceştia rulează. Agenții pot executa acțiuni proactive pentru a-şi găsi scopurile şi pentru a-şi urma crezurile care țin de situațiile pe care le întâlnesc. De asemenea agenții pot comunica activ cu alți agenți pentru a atinge obiective mai ample.

În cadrul sistemelor ambientale există o nevoie clară de suport decizional pentru agenții, pentru a forma echipe de agenți şi să-şi asume roluri în cadrul echipelor.

Putem defini un mediu inteligent ca unul care este capabil să dobândească şi să aplice cunoştinţele despre mediu şi locuitorii săi, în scopul de a îmbunătăți experiența lor în acest mediu.

Sistemele ambientale inteligente se pot clasifica în două categorii: sisteme contruite din rețele independente şi dispozitive programabile, care au functionalități restrânse şi sisteme care trebuie să producă un comportament centrat pe utilizator şi care trebuie să reacționeze la scenarii complexe într-un mod care păstrează un accent clar pe activitățiile desfăşurate de utilizatori zi de zi. Deşi suntem atraşi să credem că un comportament centrat pe utilizator va reieşii din interacțiunea dintre diverse dispozitive, exstă motive puternice care sugerează o abordare mai directă în cele mai multe cazuri. În timp ce numeroase aplicații la scară mică şi-au doveditimplică utilitatea, aplicațiile mari rămân încă evazive, deoarece este foarte greu să se anticipeze ce se întâmplă în lumea reală şi cu atât mai puțin să se traducă aceste lucruri într-un comportament adaptiv relevant, de asemenea este foarte greu să se administreze greşelile inevitabile care intervin în timpo viziune de ansamblu a răspunsurilor sistemului la mediul său în schimbare ce se execută acțiuni complexe, fără intervenția utilizatorului.

În general sistemele ambientale inteligente ridică următoarele întrebări:

1. Ce impact are calitatea scăzută, natura difuză şi eterogenă a datelor provenite de la senzori şi operația de inferență în programare ?

2. Ce implicații au atributele de la punctul 1 în creare de software middleware sau cadre de programare pentru inteligența ambientală ?

Sistemele ambientale inteligente nu se bazează doar pe de detectarea stimulilor interni sau externi mai degrabă se bazează pe valorificarea informațiilor dintr-o gamă diversă de surse interconectate. Acest lucru schimbă percepția de abordare, care parea a fi o problemă de detectare şi de raționament la una de management al informației şi mentenanță: o viziune de ansamblu a răspunsurilor sistemului la mediul său în schimbare.

## 2. Închidere bibliografică

Aplicațiile tradiționale oferă un anumit context în cadrul căruia pot să apară diferite interacțiuni. Pentru majoritatea sistemelor există un grad scăzut de incertitudine în cadrul acestor interacțiuni. Un utilizator poate selecta sau nu un meniu sau apăsa o anumită tastă. În cadrul anumitor sisteme pot exista erori datorită mesajelor pierdute, care conțin erori sau care sunt trimise repetitiv, aceste erori fiind tratate în majoritatea cazurilor in cadrul middlewareului. În majoritatea sistemelor există de asemenea erori de operare care trebuie corectate într-un fel sau altul, deobicei cu un impact minim asupra sistemului. Cu toate că unele interacțiuni pot genera erori în cadrul acestor sisteme, aceste erori sunt definite în prealabil.[2]

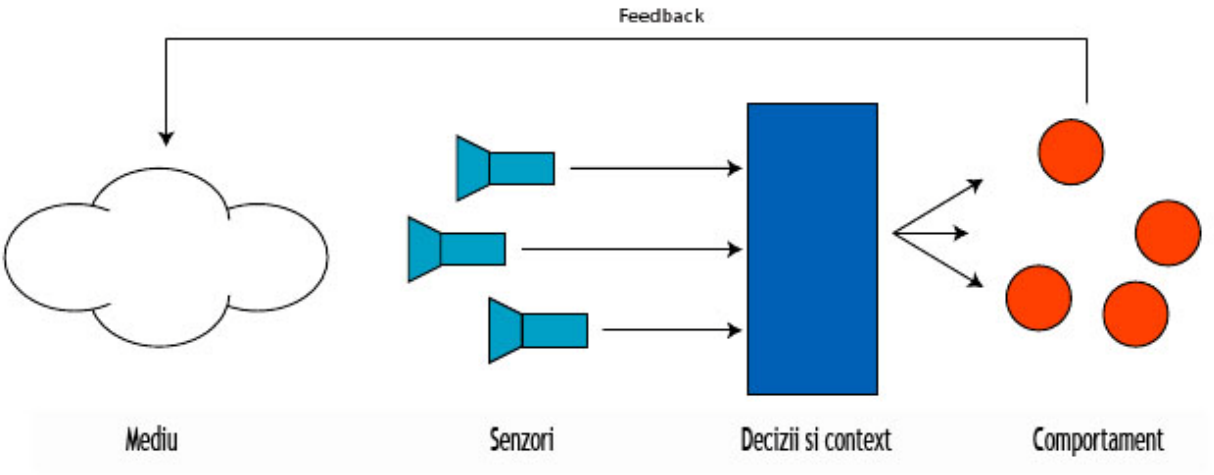


Figura 1 Modul de mapare a datelor provenite de la senzori pe un anumit comportament

Interacțiunile ambientale prezintă caracteristici complet diferite. Interacțiunile apar, în cea mai mare parte, în urma observării de către senzori a acțiunilor de zi cu zi. Chiar şi în cadrul unei singure categorii de senzori, cum ar fi cei de locație, fiecare categorie de senzori are de cele mai multe ori caracteristici radical diferite. De exemplu, un sistem Wifi de localizare, cum ar fi PlaceLab [3], poate raporta în general poziția unui dispozitiv cu precizie de 10 m, dar s-ar putea din cauza reflexie ca senzorul să raporteze o locație diferită față de locația fizică în care se află dispozitivul. Aceste condiții se pot aplica analog şi la celalalte tipuri de senzori. O aplicație proiectată după modelul din Figura 1 ar putea să execute următoare instrucțiune: “Când A şi B sunt în camera X execută acțiunea 1 ”, nici un senzor nu poate să ofere informații precise despre locație pentru a fi luată o decizie cu încredere.

Problema de bază este că nu putem atribui instrucțiuni la o singură acțiune, ci doar la un anumit număr de acțiuni intreprinse în context.

Sistemele orientate pe evenimente rezolvă această problemă prin oferirea unui model de sarcini care este implementat de aplicație. Evenimentele primite sunt folosite pentru a executa diferite sarcini. Lucrurile devin şi mai complicate când luăm în considerare sistemele ambientale care conțin mai multe aplicații sau servicii. O anumită acțiune poate fi interpretată de mai multe aplicații simultan, fiecare dintre care poate prezenta apoi un comportament. Aceste comportamente sunt independente, cu toate acestea comportamentele pot fi şi dependente, un anumit comportament fiind afectat negativ de alt comportament al aplicației.

Toate datele folosite într-un sistem ambiental inteligent pot fi considerate nesigure, acestea ridicând probleme pe care tehnologia actuală de construire a senzorilor nu le poate rezolva. Relațiile dintr datele provenite de la senzori şi scenariile suportate de sistemele ambientale nu pot fi definite exact, deoarece de cele mai multe ori unei acțiuni nu îi poate fi atribuit fără echivoc, un rol particular într-un proces care face obiectul incertitudinilor acțiunilor de zi cu zi ale utilizatorilor. Acest lucru implică faptul că orice decizie luată în timpul execuției unei sarcini poate fi greşită, în măsura în care datele au fost greşit interpretate.

Cu toate că senzorii sunt imprecişi individual, un mediu care dispune de un număr mare de senzori poate compensa aceste deficiențe individuale şi poate realiza o estimare apropiată de valoarea reală sau chiar exactă. Acest lucru poate ridica o problemă legată de numărul de senzori, existenți într-un mediu. [6].

Sistemele ambientale pot îmbunătății estimările datelor, folosindu-se de date citite în trecut, de valorile implicite şi de datele citite de la senzori. Un punct important care trebuie luat în considerare este decuplarea comportamentului de observațiile făcute izolat şi de a folosi în schimb un model global al mediului, care poate fi determint de toate aspectele mediului care pot fi detectate de senzori. Pentru a evita ca datele eronate să fie propagate în sistem, trebuie luate în considerare următoarele:

- Evenimentele pot propaga erori, drept urmare nu pot servi ca bază pentru programare. Un eveniment provenit de la un senzor, cum ar fi poziția unui individ într-un anumit spațiu, poate fi eronat. Aplicația va reacționa la acest eveniment declanşat de date eronate, lucru care nu se doreşte, drept urmare aplicația trebuie să identifice datele eronate local, înainte de a executa acțiuni.
- Evitarea acțiunilor asupra datelor provenite de la un singur senzor. Pentru a evita datele eronate, acțiunile trebuie luate pe baza datelor preluate de la mai multe surse.
- Interconectarea este mai importantă decât datele. Nu putem considera un sistem ambiental inteligent în cazul în care este pur şi simplu condus de evenimente provenite

de la senzori, neținând cont şi de alți factori. Modelul, relațiile sale şi capacitatea de a extrage informații prin intermediul fuziunii caracterizează sistemul mai mult decât disponibilitatea senzorilor.

- Orice decizie are nevoie de o strategie de atenuare. Este inevitabilă apariția erorilor, iar aceste erori pot cauza probleme majore pentru aplicație dacă nu sunt luate în calcul la proiectarea sistemului.
- Lucrurile intersante vin din compoziție. Compunerea aplicațiilor şi serviciilor au impact asupra întregului sistem, de aceea rezultatele compoziției trebuie să fie definite din timp.

În proiectarea unui sistem abiental inteligent ar trebui să se țină cont de următoarele procese: percepție, raționament şi actiune. Aceste trei procese trebuie să interacționeze continuu pentru a realiza un sistem ambiental complet.

Procesul de percepție ar trebui să fie împărțit în diferite sarcini, în scopul de a oferi o percepție corectă asupra lumii reale. Acesta trebuie să se ocupe cu detalii low-level pentru a prelua datele de la lumea reală şi să le adapteze la o bază de cunoştințe. Acest proces trebuie să şteargă datele eronate, nesemnificative sau valorile redundante în scopul de a obține acuratețea necesară proceselor următoare. Procesul de percepție este împărțit în 5 etape: colectarea datelor, verificarea datelor, corectarea datelor, filtrarea datelor şi ontologizarea datelor.

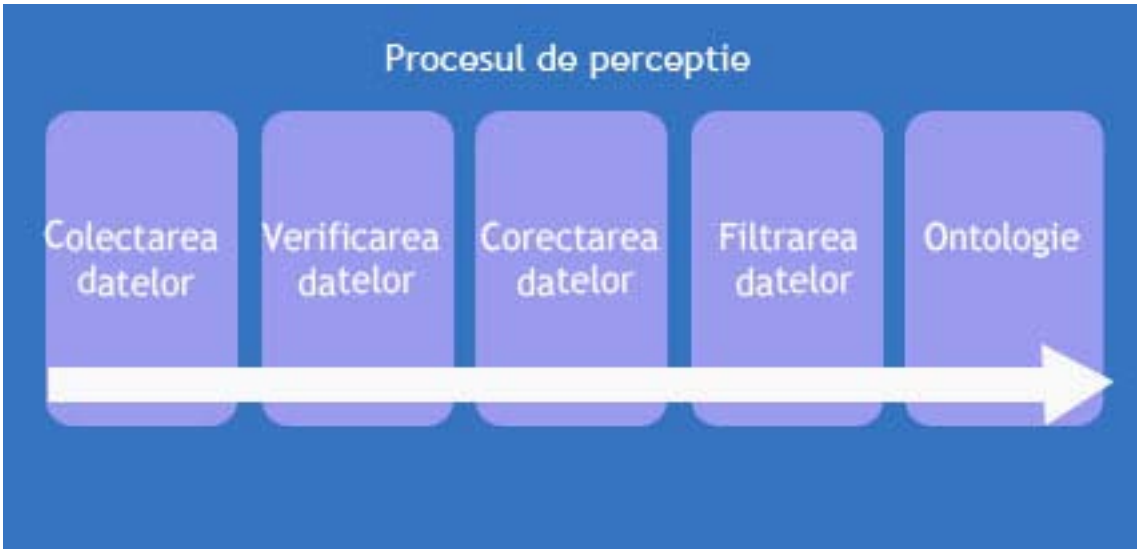


Figura 2 Procesul de percepție

Colectarea datelor este o sarcină de nivel scăzut, obictivul său principal fiind colectarea datelor de la dispozitivele fizice. Această sarcină va interacționa în general cu toate dispozitivele

plasate în mediu monitorizat de sistemului ambiental inteligent. Pentru ca sarcina de colectare a datelor să ruleze cât mai eficient senzorii trebuie amplasați cât mai organizat pentru a se evita informația redundantă sau nesemnificativă şi pentru a reduce costurile. De asemenea trebuie luat în calcul şi tipurile de dispozitive folosite, aceastea putând fi programabile sau nu. Datorită faptului că majoritatea dispozitivelor fiind programate de către producător, sarcina de colectare a datelor trebuie să fie realizată ținând cont de diferitele implementări ale dispozitivelor.

Etapa de verificarea datelor este una dintre cele mai importante etape. Scopul principal al acestei sarcini este de a verifica datele colectate în etapa de colectare a datelor. În această etapă se poate verifica dacă datele sunt corecte sau nu. Adevărata provocare din parte programatorului este să verifice corectitudinea datelor. Datorită faptului că nu există o soluție universală pentru toate mediile existente, sarcina de verificare trebuie să se adapteze în mod dinamic la fiecare mediu.

În cadru acestei sarcini se poate introduce un motor de reguli, care poate rula reguli de verificare pentru a determina dacă datele sunt corecte sau nu pentru mediul curent. Motorul de reguli trebuie să le ofere programatorilor o modalitate flexibila de a adăuga, modifica sau şterge reguli.

Această sarcină trebuie să ruleze în strânsă legătură cu sarcina de verificare a datelor. Sarcina de verificare a datelor trebuie să poată invoca în mod direct sarcina de corectare.

Verificarea datelor poate fi privită ca un filtru aplicat asupra datelor primite care poate fi folosit să accepte date sau să le respingă din diferite motive (incorecte, redundante sau alte motive adăugate de programatori.

Corectarea datelor poate fi realizată  în diferite moduri:

- Ignorarea datelor - acest lucru presupune setarea unei valori implicite ( ex: valoarea null pentru SQL).
- Ajustarea datelor - se poate modifica datele atribuindu-le ultima valoare corectă înregistrată.
- Respingerea datelor - dacă nu se poate poate aplica nici una dintre metodele de mai sus datele pot fi respinse.

Algoritmi de inteligență artificială au nevoie de o bază solidă de cunoştiințe pentru a funcționa. Scopul acestei sarcini este acela de a organiza datele conform unui model din lumea reală. Această sarcină va trebui să sincronizeze toate datele care provin dintr-un mediu populat cu dispozitive şi evenimente asincrone. Sistemele ambientale inteligente au nevoie uneori de seturi de date compuse din valorile citite de la mai multe dispozitive. Implementarea unui

proces de sincronizare poate varia, dar majoritatea sistemelor inteligente au în comun următoarele elemente:

- Un buffer de date.
- Un garbage collector.

După ce datele sunt agregate acestea pot fi stocate într-o bază de cunoştiințe care va fi accesată de sarcina care se ocupă de raționament sau chiar sarcina care se ocupă de procesul de învățare.

În cadrul sistemelor inteligente procesul de raționament poate fi împărțit în trei sarcini distincte: învățare, raționament şi predicție. Procesul de învățare poate fi privit ca un liant pentru celălalte sarcini care au ca scop procesul de raționament. Pentru realizarea procesului de învățare se pot folosi două familii de algoritmi specifici pentru sistemele inteligente care integrează acest proces. Cele două familii de algoritmi sunt SVM (support vector machine) şi NN (neutral networks). Cel mai mare avantaj al acestor algoritmi este că acestia returnează tot timpul date la sfărşitul execuției, pe de altă parte aceste date sunt greu de înteles. Procesul de învățare se poate baza de asemenea pe data mining, proces care se ocupă cu extragerea informațiilor şi găsirea anumitor tipare din datele salvate în baza de cunoştiințe.

O situație este definită ca un set de stări care au valori similare. Un set poate fi definit de:

- o stare canonică reprezentativă pentru toate stările din set.
- o funcție care defineşte cum se pot măsura diferențele dintre două stări.
- un prag care defineşte diferența maximă dintre starea canonică şi celălalte stări din set.
- o valoare pentru fiecare element al unei stări din intervalul [0,1]. Valoarea 0 indicând faptul că starea respectivă poate fi ignorată.

Cunoştiințele dobândite în urma procesului de învățare sunt utile pentru a face predicții despre următoarele acțiuni care pot fi luate în calcul pentru a îndeplini automat unele obiective. Aceste acțiuni pot fi executate pe baza datelor de ieşire de la algoritmi specifici sistemelor inteligente. Exemplu de predicții:

- Prezicerea evenimentelor şi stărilor sistemului.
- Prezicerea acțiunilor locuitorilor mediului.
- Prezicerea anomaliilor.

Scopul sarcinii de detectare a erorilor este de a supraveghea acțiunile luate de locuitori, pentru a detecta decizii opuse între sistemul ambiental şi locuitori. Prin urmare ultimele acțiuni luate de sistemul inteligent trebuie memorate şi comparate cu acțiunile luate de locuitori. Compararea acestor acțiuni presupune de obicei operațiuni complexe la care iau parte multe

elemente componente ale sistemului şi de asemenea trebuie luat în considerare felul în care aceste acțiuni sunt comparate, durata de timp alocată pentru comparare, sau dacă acțiuni asupra diferitelor componente pot fi considerate opuse. Toate aceste aspecte fac detectarea acțiunilor opuse mult mai provocatoare.

Sistemele ambientale inteligente trebuie să acționeze independente pentru a fi considerate sisteme inteligente. Deciziile şi sarcinile ordonate de procesul de raționament, trebuie să treacă prin trei sarcini principale ale acestui proces:

Policy manager - decizia executării unei sarcini trebuie supervizată de acest proces. Un sistem inteligent trebuie să definească parametrii pentru economisirea energiei, securitate şi comfort. Acest proces trebuie să poată modifica parametrii în timpul execuției, deoarece preferințele locuitorilor pot varia.

Planificatorul de sarcini - acest proces trebuie să poată planifica sarcinile în funcție de timpul de execuție sau de prioritate.

Procesul care execută sarcini - acest proces trebuie să interacționeze cu protocoale low level pentru a trimite instrucțiuni dispozitivelor.

## 3. Plan de lucru

Lucrarea va fi impărțită în trei părți, după cum urmează

Partea I: Modele multi-agent pentru sisteme ambientale inteligente.

- Capitolul 1. Ingineria sistemelor Ambientale Inteligente [11]
- Capitolul 2. Sisteme ambientale multi-agent [12]
- Capitolul 3. Metodologii de dezvoltare a sistemelor multi-agent. [14]

Partea II: Arhitectura multi-agent a middleware-ului pentru senzori.

- Capitolul 1. Modelul arhitectural blackboard [13]
- Capitolul 2. Tipuri de agenti
  - 2.1 Agenti BDI
  - 2.2 Agenti bazati pe reguli
  - 2.3 Agenti planificatori
- Capitolul 3. Descrierea modelului arhitectural

Partea III: Proiectarea şi dezvoltarea middleware-ului.

## 4. Bibliografie


1. Dobson, S., Nixon, P.: More principled design of pervasive computing systems. In: Bastide, R., Palanque, P., Roth, J. (eds.) Human computer interaction and interactive systems. LNCS, vol. 3425, Springer, Heidelberg (2005)

2. Kephart, J., Chess, D.: The vision of autonomic computing. IEEE Computer 36, 41–52 (2003)

3. Hightower, J., LaMarca, A., Smith, I.: Practical lessons from PlaceLab. IEEE Pervasive Computing 5, 12–19 (2006)

4. Pinhanez, C., Bobick, A.: Human action detection using PNF propagation of temporal constraints. In: Proceedings of CVPR'98, pp. 898–904 (1998)

5. Allen, J., Ferguson, G.: Actions and events in interval temporal logic. Journal of Logic and Computation 4, 531–579 (1994)

6. Dobson, S.: Leveraging the subtleties of location. In: Bailly, G., Crowley, J., Privat, G., eds.: Proceedings of Smart Objects and Ambient Intelligence, pp. 175–179 ( 2005)

7. Hayton, R., Bacon, J., Bates, J., Moody, K.: Using events to build large-scale distributed applications. In: Proceedings of the 7th ACMSIGOPS European workshop on systems support for worldwide applications, ACM Press, pp. 9–16. ACM Press, New York (1996)

8. Clear, A.K., Knox, S., Ye, J., Coyle, L., Dobson, S., Nixon, P.: Integrating multiple contexts and ontologies in a pervasive computing framework. In: Contexts and ontologies: theory, practice and applications (2006)

9. Coyle, L., Neely, S., Rey, G., Stevenson, G., Sullivan, M., Dobson, S., Nixon, P.: Sensor fusion-based middleware for assisted living. In: Nugent, C., Augusto, J.C. (eds.) Smart homes and beyond, pp. 281–288. IOS Press, Amsterdam (2006)

10. Dobson, S., Coyle, L., Nixon, P.: Hybridising events and knowledge as a basis for building autonomic systems. Journal of Autonomic and Trusted Computing, To appear ( 2007)

11. Dobson, S., Coyle, L., Nixon, P.: Whole-System Programming of Adaptive Ambient Intelligence. Systems Research Group, School of Computer Science and Informatics UCD Dublin IE. 2008

12. Kendall E. Nygard, Dianxiang Xu, Jonathan Pikalek, Multi-agent Designs for Ambient Systems, 2008

13. Otavio Rezende da Silva, Alessandro Fabricio Garcia, Carlos José Pereira de Lucena. The Reflective Blackboard Architectural Pattern. 2002

14. Michael Wooldridge. Agent-Based Software Engineering, 1997.

15. Shung-Bin Yan , Zu-Nien Lin, Hsun-Jen Hsu, Feng-Jian Wang. Intention Scheduling for BDI Agent Systems. 2005