



UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI
MINISTERUL MUNCII, FAMILIEI
ȘI PROTECȚIEI SOCIALE
AMPOSDRUFondul Social European
POSDRU 2007-2013Instrumente Structurale
2007-2013

OIPOSDRU

UNIVERSITATEA "POLITEHNICA"
din BUCUREȘTI

FONDUL SOCIAL EUROPEAN

Investește în oameni!

Programul Operațional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013

Proiect POSDRU/6/1.5/S/16 – *Doctoranzi în sprijinul inovării și competitivității*

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI

Facultatea de Automatică și Calculatoare
Catedra de Calculatoare

UNIVERSITATEA PIERRE ET MARIE CURIE (PARIS VI)

Ecole Doctorale EDITE de Paris
Laboratoire d'Informatique de Paris 6

Nr. Decizie Senat 212 din 30.09.2011

TEZĂ DE DOCTORAT*- Rezumat -**Un sistem multi-agent dependent de context
pentru medii de Inteligență Ambientală**A Context-Aware Multi-Agent System for Aml Environments***Autor:** Ing. Andrei Olaru**COMISIA DE DOCTORAT**

Președinte	Prof. dr. ing. Dumitru POPESCU	de la	Universitatea Politehnica București
Conducător de doctorat-1	Prof. dr. ing. Adina Magda FLOREA	de la	Universitatea Politehnica București
Conducător de doctorat-2	Prof. dr. ing. Amal EL FALLAH SEGHROUCHNI	de la	Universitatea Pierre et Marie Curie (Paris VI)
Referent	Prof. dr. ing. Costin BĂDICĂ	de la	Universitatea din Craiova
Referent	Prof. dr. ing. Salima HASSAS	de la	Universitatea Claude Bernard (Lyon 1)
Referent	Prof. dr. ing. Cristian GIUMALE	de la	Universitatea Politehnica București
Referent	Prof. dr. ing. Nicolas MAUDET	de la	Universitatea Pierre et Marie Curie (Paris VI)

București 2011

Rezumat

Inteligența Ambientă – Ambient Intelligence, sau AmI – este viziunea unui mediu electronic omniprezent (ubicuu), care este non-intruziv, dar, de asemenea, pro-activ, și care ajută oamenii într-o manieră personalizată și dependentă de context, în sarcinile lor de zi cu zi. Cele mai multe implementări de sisteme de inteligență ambientă realizate în ultimul deceniu s-au concentrat pe punerea în aplicare de sisteme complete care servesc unor scopuri specifice. În această cercetare ne concentram pe stratul unui sistem pentru AmI care gestionează schimbul de informații bazat pe semantica lor, în scopul de a furniza informații relevante utilizatorului interesat.

Această teză prezintă efortul de cercetare către realizarea unui sistem multi-agent pentru Inteligența Ambientă, care asigură fluxul de informații dependent de context, și în care dependența de context este integrată astfel încât agenții să gestioneze și să schimbe informații context în mod natural, într-un mod generic. Abordarea noastră de a construi un sistem multi-agent dependent de context pentru Inteligența Ambientă se bazează pe trei aspecte: o reprezentare bazată pe grafuri pentru informațiile de context, care este cuplată cu definiția de șabloane de context pentru recunoașterea situației; o topologie a sistemului bazată pe context, în care relațiile de vecinătate reflectă existența unui context partajat între agenți; precum și un comportament local pentru agenți, care se bazează pe mecanisme de auto-organizare, în scopul de a oferi rezultate coerente la nivel global.

Cercetarea teoretică care este prezentată în această lucrare a fost validată prin intermediul a trei proiecte: AmIciTy:Mi se ocupă cu răspândirea de informații într-un sistem format dintr-un număr mare de agenți; prototipul Ao Dai demonstrează corespondența între structura contextului și ierarhiile de agenți; și platforma Ao Dai, care este un sistem multi-agent pentru implementarea de aplicații AmI.

Contents

Introducere	3
Contextul tezei	4
Motivație	4
Obiective	5
Structura tezei	6
1 Definirea problemei	8
1.1 Viziunea inteligenței ambiante, sau AmI	8
1.1.1 Scenarii	9
1.1.2 Caracteristici	9
1.1.3 Aplicații	10
1.1.4 Niveluri	10
1.2 Ce înseamnă cu adevărat AmI?	11
1.3 Câteva noi scenarii	11
1.4 Elemente ale abordării	12
1.5 Stabilirea scopurilor pentru această cercetare	13
2 Un studiu al domeniilor de cercetare conexe	14
2.1 Medii de AmI bazate pe agenți	14
2.2 Dependența de context	15
2.3 Emergență și sisteme cu auto-organizare	16
3 Comportamentul agenților: folosirea auto-organizării	18
3.1 Abordare	18
3.2 AmIciTy:Mi	19
3.3 Evaluare	21
4 Structurarea sistemului multi-agent	23
4.1 Abordare	23
4.2 The Ao Dai Project	24

5	Îmbunătățirea dependenței de context	27
5.1	O abordare holistică pentru dependența de context	28
5.2	Dependență de context în interiorul agentului	29
5.3	Dependența de context în exteriorul agentului	31
5.4	Un comportament îmbunătățit pentru agenți	32
6	O nouă platformă pentru aplicații AmI	34
6.1	Experimente	38
7	Concluzii	40
7.1	Perspective	42
	Lista de publicații	44

Introducere

Inteligența Ambientă – Ambient Intelligence, sau AmI – este viziunea unui mediu electronic omniprezent (ubicuu), care este non-intruziv, dar, de asemenea, pro-activ, și care ajută oamenii într-o manieră personalizată și dependentă de context, în sarcinile lor de zi cu zi. Acesta este integrat atât de mult în viața de zi cu zi, încât devine "invizibil". Este ușor de utilizat, de încredere, și transmite informații pentru a face ca ele să ajungă acolo unde este nevoie.

Cele mai multe implementări de sisteme de inteligență Ambientă realizate în ultimul deceniu s-au concentrat pe punerea în aplicare de sisteme complete care servesc unor scopuri specifice. În munca noastră ne concentram pe stratul unui sistem de AmI care se ocupa de schimbul de informații, conștient de semantica lor, în scopul de a furniza informații relevante pentru utilizatorul interesat. Acest strat prezintă cercetătorilor un mare număr de provocări, și este deosebit de interesant din punctul de vedere al inteligenței artificiale și al sistemelor multi-agent.

Această teză prezintă efortul de cercetare către realizarea unui sistem multi-agent pentru Inteligența Ambientă, care asigură fluxul de informații dependent de context, și în care dependența de context este integrată astfel încât agenții să gestioneze și să schimbe în mod natural informații de context, într-un mod generic, rămânând în același timp flexibil în ceea ce privește capacitățile necesare de calcul, și de asemenea păstrând posibilitatea de a integra procese specifice aplicațiilor, dacă este necesar.

Abordarea noastră de a construi un sistem multi-agent dependent de context pentru Inteligența Ambientă se bazează pe trei aspecte: o reprezentare bazată pe grafuri pentru informațiile de context, care este cuplată cu definiția de șabloane de context pentru recunoașterea situației; o topologie a sistemului bazată pe context, în care relațiile de vecinătate reflectă existența unui context partajat între agenți; precum și un comportament al agent care integrează celelalte două componente și care se bazează pe mecanisme de auto-organizare, în scopul de a oferi rezultate coerente la nivel global – nivelul sistemului – în timp ce agenții lucrează doar la nivel local.

Cercetarea teoretică care este prezentată în această lucrare a fost validată prin intermediul a trei proiecte: AmIciTy:Mi se ocupă cu răspândirea de informații într-un sistem format dintr-un număr mare de agenți simpli, dar cognitivi, controlată prin măsuri simple de context; prototipul Ao Dai demonstrează corespondența între structura contextului și ierarhiile de agenți; și platforma Ao Dai, folosind JADE și

limbajul de programare orientat-agent S-CLAIM, care este un sistem multi-agent pentru implementarea de aplicații AmI.

Contextul tezei

Termenul de inteligență ambientă a fost introdus la începutul secolului 21, în 2001, de raportul grupului ISTAG [Ducatel et al., 2001], atunci când a devenit una dintre prioritățile în domeniul ITC în Uniunea Europeană și în întreaga lume.

Inteligența ambientă ar trebui să reprezinte al treilea val în informatică. După main-frame și calculatorul personal, în epoca de inteligenței ambiante dispozitivele devin invizibile, fiind integrate în toate obiectele și materialele. Acest lucru face ca totul să devină "inteligent" și, prin comunicare, totul în jurul nostru va colabora în scopul de a oferi funcții mai complexe și rezultate mai utile. AmI reprezintă, de asemenea, o evoluție a ceea ce este acum Internetul: servicii bazate pe web, colaborative și sociale care asistă utilizatorul în activitățile de zi cu zi [Ducatel et al., 2001, Weiser, 1995].

Inteligența Ambientă este un domeniu vast, care a dat naștere mai multor direcții de cercetare și dezvoltare. Am putea considera că un sistem de AmI conține mai multe niveluri: nivelul de hardware, nivelul de rețea / interconectivitate, un nivel de interoperabilitate, nivelul aplicațiilor / serviciilor inteligente, și interfețele inteligente cu utilizatorul. Această lucrare se concentrează pe construirea de soluții pentru nivelul aplicație al Inteligenței Ambiante, mai precis pe proiectarea și construirea unui sistem multi-agent pentru AmI care se integrează dependența de context, astfel încât agenții să gestioneze și să schimbe în mod natural informații context.

Această teză are loc în contextul unei colaborări de lungă durată între Prof. Amal El Fallah Seghrouchni și prof. Adina Magda Florea, care a dus deja la realizarea tezei de doctorat a lui Alexandru Suna, mai multe schimburi internaționale pentru studenți de master și proiectul FP7 ERRIC.

Această teză este în cotutelă între Universitatea Pierre et Marie Curie și Universitatea Politehnica din București. Este finanțată prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013 al Ministerului român al Muncii, Familiei și Protecției Sociale, prin acordul financiar POSDRU/6/1.5/S/16, de Laboratorul de Informatică al Paris 6 (LIP6), și de către Agenția Universitară a Francofoniei.

Motivație

Multe implementări de sisteme pentru inteligența ambientă până în prezent au încercat să realizeze sisteme complete de AmI, care conțin elemente din toate nivelurile, în special hardware, aplicație și interfață [Augusto et al., 2010, Cook et al., 2009].

Dar a construi hardware-ul, middleware-ul, interfețele și aplicațiile duce la pierderea generalității, sau a puterii de reprezentare, sau a flexibilității și scalabilității.

Acesta este motivul pentru care am ales să ne concentrăm pe un singur nivel al unui sistem de AmI: nivelul care lucrează cu informații la nivel semantic, și care este esențial pentru dependența de context a întregului sistem.

Ca mulți alți cercetători, am ales paradigma orientată agent pentru modelare și implementare, pentru că agenții oferă caracteristici care sunt importante pentru AmI, în special autonomia și pro-activitatea [Ramos et al., 2008].

În scopul de a face acest sistem multi-agent să se comporte ca un middleware pentru o gamă mai largă de sisteme AmI viitoare, o prioritate a fost de a menține sistemul generic (spre deosebire de specific unui domeniu sau aplicație). Cerințele de disponibilitate și de încredere cer, de asemenea, distribuirea sistemului, precum și posibilitatea agenților de a lucra singuri sau în organizații de diferite dimensiuni. Mai mult decât atât, execuția agenților pe dispozitive cu o gamă largă de capacități – de la senzori la stații de lucru – a însemnat că agenții trebuie să utilizeze reprezentări și algoritmi flexibili și adaptabili.

Obiective

Problema de cercetare la care această lucrare va încerca să răspundă este ”**Cum trebuie construit un sistem multi-agent pentru nivelul aplicație al Inteligenței Ambiante?**”.

Ca rezultat al cerințelor prezentate în secțiunea precedentă, mai multe **obiectivele principale** au fost definite pentru această cercetare:

- dezvoltarea unui model de sistem multi-agent pentru inteligența ambiantă, care dispune de auto-organizare, dependență de context și anticipare;
- crearea mai multor scenarii care accentuează cerințele unui mediu de inteligență ambiantă la scară reală;
- dezvoltarea unui mediu de simulare care pune în aplicare elemente ale scenariilor, pentru experimente cu platforme AmI;
- implementarea și experimentarea cu modelul dezvoltat, folosind mediul de simulare, pentru a dovedi valabilitatea modelului ca o componentă a unui mediu inteligență ambiantă.

Realizarea acestor obiective este prezentată pe parcursul acestei lucrări. O imagine detaliată cu privire la diferitele părți și capitole din această lucrare este prezentată în secțiunea următoare.

Structura tezei

Scopul primului capitol – **Capitolul 1 "Definirea problemei"** – este de a afirma problema de cercetare pe care încercăm să o rezolvăm în restul acestei lucrări. În prima secțiune a capitolului vom examina în detaliu conceptul de inteligență ambientă, privind scenariile relevante din literatura de specialitate, aplicații ale conceptului, și caracteristicile care se așteaptă să fie furnizate de către AmI. Apoi, propunem câteva scenarii noi, care sunt îndreptate mai puțin către modul în care AmI va fi văzută de către utilizator, și mai mult către modul în care un sistem AmI ar trebui să lucreze în interior și ce cerințe ar trebui să îndeplinească. După o scurtă trecere în revistă a unor paradigme care sunt buni candidați pentru punerea în aplicare a AmI, cum ar fi sistemele multi-agent, dependența de context, precum și utilizarea de auto-organizare, vom afirma obiectivele acestei cercetări, în ultima secțiune, lăsând alte capitole ale lucrării să detalieze modul în care aceste obiective au fost îndeplinite.

Capitolul 2 "Un studiu al domeniilor de cercetare conexe" examinează lucrările legate de cercetarea noastră: utilizarea de agenți software și sisteme multi-agent în implementarea inteligenței ambiante; dependența de context și reprezentarea contextului; și în cele din urmă de auto-organizarea în sistemele de agenți cognitivi.

În construirea unui sistem multi-agent pentru inteligența ambientă, primul aspect pe care ne-am concentrat a fost comportamentul agenților – **Capitolul 3 "Comportamentul agentului: folosirea auto-organizării"**. Mai precis, modul în care ar trebui să facă schimb de informații agenții, astfel încât informațiile interesante să ajungă la agenții interesați, fără un control centralizat. După discutarea motivației pentru această abordare, vom prezenta middleware-ul AmIciTy:Mi pentru aplicații AmI, o implementare 'proof-of-concept' care demonstrează modul în care agenții se pot baza doar pe cunoștințe și interacțiuni locale pentru schimbul de informații, dar pot obține un rezultat la nivel global, la nivelul sistemului, în care răspândirea de informații poate fi ușor controlată prin intermediul unor măsuri de context simple.

Accentul se mută de la comportamentul agenților, către topologia sistemului în **Capitolul 4 "Structurarea sistemului multi-agent"**, în care se cercetează corespondența între structura ierarhică a unor aspecte ale contextului, și ierarhia logică a agenților. Prototipul Ao Dai este implementat în limbajul de programare orientat-agent CLAIM și atribuie un agent fiecărui element al contextului. Contextul dinamic este suportat prin ierarhii mobile de agenți – un agent poate deplasa cu ușurință, împreună cu întregul său context.

Dependența de context este în centrul a ceea ce este probabil capitolul cel mai important al acestei lucrări – **Capitolul 5 "Îmbunătățirea dependenței de context"**. Capitolul conține modelul formal pentru o abordare holistică a dependenței de context. Elementele pentru care AmIciTy:Mi utilizează abordări simplificate – topologia sistemului și reprezentarea contextului – sunt îmbunătățite pentru un model realist și mai puternic. În interiorul agentului, informația de context este reprezentată

de graful conceptelor care sunt relevante pentru starea actuală a agentului. Interesele agentului sunt definite ca un set de șabloane de context, care sunt grafuri cu elemente generice care se potrivesc cu o gamă mai largă de situații. În afara agentului, topologia reflectă contextul agentului, mai precis relațiile de vecinătate cu alți agenți reflectă contextul comun cu aceștia. Această abordare se bazează pe prototipul Ao Dai, dar suportă o gamă mai largă de aspecte ale contextului. Legătura dintre cele două părți – interiorul și exteriorul agentului – se face prin comportamentul agenților, care este foarte asemănător cu comportamentul agenților AmIciTy , dar de data aceasta folosește grafurile și șabloanele de context și noua topologie a sistemului.

Rezultatul concret al acestei cercetări, precum și punerea în aplicare a conceptelor prezentate în capitolele anterioare, au condus la dezvoltarea unei platforme originale – **Capitolul 6 ”O nouă platformă pentru aplicații AmI”**. Motivația pentru aceasta platforma a fost necesitatea pentru o suită de testare adecvată pentru nivelul aplicație a inteligenței ambiante. Platforma folosește framework-ul pentru dezvoltarea de agenți JADE, utilizează S-CLAIM ca limbaj de programare orientat-agent – o versiune simplificată și îmbunătățită a CLAIM – și are instrumente de vizualizare centralizată și de urmărire a sistemului multi-agent.

Ultimul capitol al tezei – **Capitolul 7 ”Concluzii”** – sintetizează realizările acestei lucrări. Ce a fost realizat este, mai mult decât orice, cercetarea cu privire la modul de a construi un sistem multi-agent pentru a servi ca middleware pentru Inteligența Ambiantă, la nivel aplicație. Unele potențiale ținte pentru viitor sunt prezentate în ultima secțiune a acestei lucrări.

Chapter 1

Definirea problemei

Scopul acestui capitol este de a stabili problema de cercetare de stat care încercăm să fie rezolvată în restul acestei lucrări. În prima secțiune a acestui capitol vom examina în detaliu conceptul de inteligență ambientă, examinând scenariile cele mai relevante din literatura de specialitate, aplicații ale conceptului, și caracteristicile care se așteaptă să fie furnizate de către Inteligența Ambientă. Aceste caracteristici vor fi în continuare clasificate și discutate în mod individual în secțiunea 1.2.

Bazat pe o mai bună înțelegere a caracteristicilor AmI, propunem unele scenarii noi, care sunt îndreptate mai puțin către modul în care AmI va fi văzută de către utilizator, și mai mult către modul în care un sistem AmI ar trebui să lucreze în interior și cerințele pe care ar trebui să le îndeplinească (Secțiunea 1.3).

După o scurtă trecere în revistă a paradigmatelor care sunt buni candidați pentru implementarea AmI, cum ar fi sistemele multi-agent, dependența de context, precum și utilizarea de auto-organizare, vom stabili obiectivele acestei cercetări, în ultima secțiune (1.5), urmând ca în celelalte capitole din această lucrare să fie detaliat modul în care aceste obiective au fost îndeplinite.

1.1 Viziunea inteligenței ambiante, sau AmI

Inteligența Ambientă este un concept care a fost introdus la sfârșitul anilor 1990, și este un mediu omniprezent electronic care este non-intruziv, dar, de asemenea, proactiv, și care ajută oamenii, într-o manieră personalizată și dependentă de context, în sarcinile lor de zi cu zi. În timp ce noțiunea de inteligență ambientă a fost introdusă în 2001 de către raportul ISTAG [Ducatel et al., 2001], ea se bazează pe conceptul mai vechi de calcul ubicuu (Ubiquitous Computing) introdus de Weiser un deceniu mai devreme [Weiser, 1995].

Ambient Intelligence ar trebui să reprezinte al treilea val în informatică, după main-

frame și calculatorul personal, prin integrarea puterii de calcul în mediul înconjurător, și, în loc de a folosi interfețe dedicate pentru a interacționa cu utilizatorul, ar trebui să utilizeze obiecte de zi cu zi înzestrate cu funcționalități AmI. Acest lucru va face computerele "invizibile" și, prin folosirea obiectelor de zi cu zi ca interfață, mai intuitive și naturale în utilizare.

Pentru munca noastră, un aspect esențial într-un adevărat sistem de Inteligență Ambientă este informația, și de modul în care informația se deplasează prin intermediul sistemului, între dispozitive, precum și între utilizatori – Weiser numește asta "o lume de transportoare de informații". Această lucrare propune mecanisme și reprezentări care asigură că aceste informațiile vor fi schimbate și livrate într-un mod care este relevant, dependent de context, și inteligent.

1.1.1 Scenarii

Această viziune a fost descrisă în domeniul de Inteligenței Ambiente, în principal prin intermediul unor scenarii. Acestea sunt povești în care personajul(e) principal(e) utilizează și interacționează cu sistemul de inteligență ambientă. Din scenarii putem extrage caracteristicile și funcționalitățile AmI și putem afla cum ar trebui să lucreze și să fie organizat un sistem AmI.

Vom examina câteva scenarii relevante pentru inteligență ambientă și de calcul ubicuu, iar apoi vom extrage și clasifica caracteristici care sunt descrise în aceste scenarii.

1.1.2 Caracteristici

În scenariile de AmI observăm o mulțime de caracteristici care sunt legate de interfață (pentru că asta este ceea ce utilizatorul vede și se simte): ferestrele inteligente ale lui Weiser, frigiderul inteligent, recunoașterea vorbirii, chiar și controlul vremii. Dar în timp ce interfețele sunt foarte importante, nu trebuie să uităm că informațiile pe care le afișează și / sau obțin sunt gestionate de un sistem care, chiar dacă nu este (și nu ar trebui să fie) vizibil, este foarte complex. Ne vom concentra în această secțiune pe caracteristicile care sunt mai interesante din acest punct de vedere.

AmI trebuie să fie **omniprezent**, **pervaziv**. Arhitectura sa trebuie să suporte un număr mare de dispozitive mobile care partajează neîncetat cantități mari de informații, fără știrea utilizatorului (dar nu și împotriva preferințelor utilizatorului). Dar AmI ar trebui să fie *distribuită* și să lucreze la un nivel *local*. AmI trebuie să fie **naturală**, prin utilizarea interfețelor avansate, intuitive, multi-modale. AmI trebuie să se adapteze tuturor și necesită numai atenția pe care utilizatorul este dispus să investească. AmI trebuie să fie previzibilă și transparentă, fiind capabilă de a

face utilizatorul să înțeleagă de ce informațiile și serviciile sunt acolo și modul în care sistemul funcționează, cel puțin în principiu. Acest lucru înseamnă că principiile de bază după care lucrează AmI ar trebui să fie *simple* și ușor de înțeles de către oricine, și de asemenea să facă AmI *generică* și *adaptivă*. Dar AmI trebuie să fie, de asemenea, **pro-activă** și inteligentă. Aceasta trebuie să adopte măsuri adecvate, fără să deranjeze. Acțiunea trebuie să fie luată doar în cazul în care utilizatorul ar înțelege cauzalitatea, și numai în cazul în care utilizatorul ar aproba acțiunea. Acesta este un caz în care *dependența de context* are un rol principal. Legat de ultimele afirmații este, de asemenea, **trasabilitatea** AmI. Trasabilitatea îmbunătățește semnificativ acceptarea de către utilizator, deoarece utilizatorii sunt mult mai toleranți față de acțiunile incorecte făcute de aplicații dependente de context în cazul în care sunt capabili să înțeleagă că au o bază rațională. AmI trebuie să furnizeze un flux de informații **previzibil**, natural. Mai este o problemă foarte importantă legată de dependența de context și AmI: același sistem de AmI va trebui să suporte **mai mult de un utilizator** la un moment dat (spre deosebire de cum vedem de obicei în scenarii). În spațiile publice există un număr mare de utilizatori pe care AmI trebuie să fie capabilă de a-i notifica și asista, fără a pierde confidențialitatea (sau informații importante), față de alți utilizatori.

1.1.3 Aplicații

Cercetarea în acest domeniu este vastă și variată. Și asta este normal, având în vedere complexitatea a ceea ce ar trebui să fie inteligență ambientă. Literatura de specialitate și studiile asupra AmI [Cook et al., 2009, Augusto et al., 2010] arată că există câteva direcții de dezvoltare care sunt legate de AmI: percepție, raționament, decizie, interacțiunea om-calculator, și viața privată și securitate.

Cu toate acestea, aceleași studii arată că ceva lipsește. Autorii observă că foarte puțin a fost făcut în domeniul raționamentului [Cook et al., 2009]. Deși ”intelența” este un cuvânt care face parte din numele domeniului, istețimea aplicațiilor este foarte specifică în majoritatea implementărilor, și nu poate fi folosită într-un mod generic în toate scenariile AmI. Mult mai mult timp trece în dezvoltarea de caracteristici atractive și interfețe decât în a face aceste funcții să fie utilizate într-un mod inteligent.

1.1.4 Niveluri

Inteligența Ambientă ar trebui să fie al treilea val în informatică. Acesta poate fi, de asemenea, văzută ca o evoluție a ceea ce este astăzi Internetul. Prin urmare, este de așteptat ca acesta va fi un sistem foarte complex, cu componente care acoperă multe domenii în dezvoltarea de software și hardware. Așa cum Internetul este astăzi, este ușor să organizăm componentele AmI pe mai multe niveluri (bazat pe o prezentare de către El Fallah Seghrouchni [El Fallah Seghrouchni, 2008]): hardware, conținând

dispozitivele din mediul AmI; rețeaua, care interconectează dispozitivele; un strat de interoperabilitate, care asigură protocoale uniforme; nivelul aplicație / servicii inteligente, oferind servicii dependente de semantică și context; și nivelul interfață, care permite comunicarea naturală între utilizator și sistem.

1.2 Ce înseamnă cu adevărat AmI?

Multe scenarii pentru Inteligența Ambientă sunt o combinație de elemente de pe mai multe niveluri ale AmI. De cele mai multe ori, acestea sunt concentrate pe două dintre straturi: interfețele inteligente și serviciile dependente de context. Uneori, problemele legate de interoperabilitate și de conectivitate sunt de asemenea menționate, dar mult mai rar (deși provocările din aceste niveluri nu sunt încă rezolvate).

Combinarea acestor elemente poate fi confuză pentru proiectantul unui sistem de AmI, care trebuie să fie implementat de către o echipă de cercetare: cum AmI este un domeniu vast, este greu pentru aceeași echipă să dezvolte funcționalități de conectivitate, de dependență de context, nivelul de interoperabilitate și interfețe noi, în același timp. Acesta este motivul pentru care majoritatea implementărilor, în timp ce încearcă să abordeze toate aceste probleme, au ca rezultat un sistem care nu este flexibil sau scalabil, sau care este specific unui domeniu de aplicație, și nu este la fel de impresionant ca scenariile descrise inițial.

În această secțiune vom discuta provocări importante în construirea sistemelor de AmI, cum ar fi **scalare, preferințele utilizatorului, debit și tehnologia care dispare, anticipare** și probleme de **viață privată și de securitate**.

1.3 Câteva noi scenarii

Această secțiune va prezenta mai multe scenarii noi, care oferă o imagine despre modul în care un sistem de inteligență ambientă poate lucra pe plan intern. Detaliile sunt axate pe nivelul aplicație al sistemului, privind modul în care informațiile sunt gestionate și modul în care deciziile sunt luate în funcție de context. Având această abordare, vom considera că utilizatorii folosesc hardware-ul de astăzi, dar dispozitivele vor fi îmbogățite cu agenți AmI, care vor forma nivelul aplicație al sistemului.

Scenariile prezentate se referă la caracteristici cum ar fi **adaptabilitatea și scalabilitatea, rezolvarea problemelor, și utilizatori multipli și colaborare**. În cele din urmă, vom prezenta scenariul pe care îl vom folosi pentru exemplele și studiile de caz din această lucrare. Acest scenariu a fost elaborat împreună cu colegii de la Honiden-Lab din Institutul NII, Tokyo, în scopul de a fi implementat în camera

inteligentă care a fost construită la Honiden-Lab. Scenariul prezintă un echilibru între funcționalități bazate doar pe informație și caracteristici AmI percepute ca fiind mai frecvente: avem detectarea și recunoașterea de persoane, precum și ecrane tactile mari; avem, de asemenea, recunoașterea contextului, lucru colaborativ și utilizarea de șabloane.

1.4 Elemente ale abordării

Sisteme multi-agent și paradigma orientată-agent

Agenții oferă caracteristici care sunt foarte necesare pentru AmI, cum ar fi reactivitate, autonomie, pro-activitate, raționament sau anticipare [Ramos et al., 2008]. De fapt, agenții oferă posibilitatea de a trece de la paradigma de calcul distribuit – în care proiectantul specifică protocolul și prelucrările așa cum sunt văzute de la nivel global – la o paradigmă bazată pe raționament local și interacțiune, în care agenții sunt proiectați din punct de vedere local iar comportamentului la nivel global este emergent.

Dependența de context

Este dificil să vorbim despre inteligență ambientă, fără a menționa dependența de context. Multe sisteme cu aplicații în Inteligența Ambientă implementează dependența de context ca una dintre caracteristicile lor de bază. În literatura relativă la dependența de context există, de obicei, două puncte de concentrare: unul este arhitectura pentru captarea informațiilor contextuale; iar celălalt este modelarea informațiilor contextuale și de modul în care se poate raționa despre context. Contextul este orice informații despre entitățile care sunt relevante pentru interacțiunea dintre utilizator și sistem [Dey, 2001]. Dependența de context este abilitatea unui sistem de a se adapta în mod autonom la contextul actual, în scopul de a oferi un răspuns și o experiență mai bune pentru utilizator.

Adăugarea un aspect de auto-organizare

Având în vedere cantitatea de dispozitive de comunicare într-un scenariu de AmI la scară reală, ar fi interesant să privim în domeniul sistemelor cu auto-organizare pentru soluții cu privire la modul în care un astfel de sistem poate să atingă un anumit scop. Emergența și auto-organizare [Heylighen, 2002] oferă mijloacele pentru a obține proprietăți complexe dintr-un număr mare de indivizi simpli care interacționează. Posibilitatea de a obține efecte noi, non-aditive ale unor interacțiuni cauzale a creat direcții de cercetare în studiul comportamentului emergent atât în ambele sisteme naturale cât și artificiale. În scopul de a modela și proiecta sisteme artificiale, sursa de inspirație au fost fie sistemele complexe de entități neînsuflețite, sau sistemele complexe de ființe vii simple (cum ar fi furnici, viespi, păianjeni, etc). Acest lucru se datorează faptului că aceste tipuri de indivizi sunt mai ușor de înțeles și de modelat.

1.5 Stabilirea scopurilor pentru această cercetare

În cadrul perspective pe niveluri pe care am expus-o, multe provocări în AmI se află la nivelul al patrulea – aplicații și servicii inteligente, ambiante. Prezenta cercetare încearcă să răspundă la unele dintre aceste provocări, prin **dezvoltarea unui sistem multi-agent pentru nivelul aplicație al unui sistem de AmI**. Soluția propusă se bazează pe o serie de caracteristici cheie: **distribuția sistemului** – un mediu fiabil de AmI trebuie să prezinte control distribuit, astfel încât funcționarea acestuia nu va fi extrem de afectată de dispariția sau indisponibilitate temporară a oricărui dispozitiv din mediu; utilizarea de **agenți software cognitivi** care au un comportament flexibil, în funcție de capacitățile dispozitivului; utilizarea de **mecanisme de auto-organizare** care să ofere mijloacele de a coordona un număr mare de agenți și de a obține, prin interacțiune locală și fără a fi nevoie de control centralizat, proprietăți globale la nivelul sistemului; **dependența de context** ca trăsătură de bază a sistemului multi-agent, care afectează atât structura sistemului de agenți cât și comportamentul agenților în scopul de a oferi o experiență optimă.

Chapter 2

Un studiu al domeniilor de cercetare conexe

Acum, că obiectivele acestei cercetări sunt clare, acest capitol examinează lucrări care sunt legate de domeniile noastre de cercetare: agenți software și sisteme multi-agent în inteligență ambientă; dependența de context și reprezentarea contextului; și în cele din urmă auto-organizare în sistemele de agenți cognitivi.

2.1 Medii de AmI bazate pe agenți

În domeniul platforme de inteligență ambientă bazate pe agenți există două direcții principale de dezvoltare: una cu privire la agenții orientați spre asistarea utilizatorului, cu funcționalități de învățare și de raționament complexe, și folosind componente centralizate (baze de cunoștințe, ontologii, etc); și una cu privire la coordonarea de agenți asociați dispozitivelor, eventual mobili, într-un context de funcționalitate mai simplă, luând în considerare, de asemenea, controlul distribuit și toleranța la erori.

Utilizarea de agenți puțini, mai complecși

Prima abordare de a folosi MAS pentru AmI este mai aproape de interfețele inteligente cu utilizatorul și anticiparea locală a intențiilor de utilizatorului, și provin din domeniul asistenților personali inteligenți. În această abordare, agenții sunt complecși, utilizează algoritmi de învățare, și se folosesc de componente centralizate pentru a regăsi date externe. Comunicarea inter-agentul este limitată, cu excepția când componentele centrale sunt desemnate ca unul sau mai mulți agenți. Printre proiectele de acest tip, putem menționa iDorm, MyCampus, EasyMeeting, sau platforma DALICA.

Utilizarea de mulți agenți, mai simpli

A doua abordare pentru platforme de AmI bazate pe agenți se referă la rezolvarea diferitelor probleme cum ar fi mobilitatea utilizatorului, controlul distribuit, auto-organizarea și toleranța la erori, având o perspectivă mai globală privind modul în care o platformă AmI ar trebui să funcționeze.

Platforma SpacialAgents este o arhitectură foarte interesantă care folosește agenți mobili pentru a oferi funcționalitate pe dispozitivele utilizatorului. Proiectul LAICA aduce argumente bune pentru utilizarea agenților în implementarea AmI. Infrastructura AmbieAgents este propusă ca o soluție scalabilă pentru serviciile de obținere de informații dependente de context. În cele din urmă, platforma bazată pe agenți CAMPUS consideră aspecte cum ar fi diferitele tipuri de contexte și controlul descentralizat.

Am rezumat anumite caracteristici care sunt relevante pentru activitatea noastră, așa cum sunt manifestate de sistemele pe care le-am analizat mai sus. Se poate observa că sisteme de agenți diferite iau în considerare diferite aspecte ale inteligenței ambiante și adoptă abordări diferite pentru punerea lor în aplicare – de exemplu, în ceea ce privește centralizarea sistemului. De asemenea, este demn de remarcat faptul că puține dintre sisteme abordează doar problema de middleware, multe dintre ele încercând să propună o arhitectură completă, de la nivelul de percepție la interfața cu utilizatorul.

2.2 Dependența de context

Dependența de contextul este o problemă centrală în domeniul inteligenței ambiante. Comportamentul pro-activ, dar non-intruziv nu ar fi posibil fără o înțelegere adecvată din partea sistemului de AmI a contextului utilizatorului. Acțiunile sistemului trebuie să pară naturale și corect integrate în situația actuală.

Contextul a fost definit ca ”orice informații care pot fi utilizate pentru a caracteriza situația entităților (adică persoane, locuri sau obiecte), care sunt considerate relevante pentru interacțiunea dintre un utilizator și o aplicație, inclusiv utilizatorul și aplicația înșiși” [Dey, 2001]. Prin urmare, dependența de context este nu numai abilitatea de a adapta reacția sistemului la situația actuală, dar, de asemenea, de a decide măsurile care trebuie luate în contextul utilizatorului.

Mulți autori consideră contextul ca aproape exclusiv cu privire la loc, momentul de timp, și alte proprietăți instantanee ale mediului fizic. Dar contextul este mai mult decât atât. În primul rând, există mai multe tipuri de context – de exemplu, computațional, temporal, social [Chen and Kotz, 2000]. În al doilea rând, contextul nu este numai format din tipurile de context menționate – unde se află utilizatorul, ce ora este, ce temperatură este afară și ce capacități are conexiunea la rețea – contextul este, de asemenea, definit de asociații între diverse fapte care se referă la utilizator, fapte care nu sunt neapărat informații contextuale din tipurile menționate.

Reprezentarea contextului

Contextul poate fi modelată în maniere diferite, ducând la o gama mare de reprezentări pentru informațiile de context [Bolchini et al., 2007, Perttunen et al., 2009].

Reprezentările cele mai simple utilizează perechi cheie-valoare sau tupluri pentru a reține măsuri pentru diverse aspecte de context, cum ar fi, de exemplu, poziția utilizatorului. Formalismele bazate pe logic folosesc mecanisme care provin din inteligența artificială și reprezentarea cunoștințelor. Mai mult folosite printre acestea sunt logicile de descriere și ontologiile. Deși ontologiile permit reprezentări de concepte complexe și constrângeri, ele nu sunt adecvate pentru reprezentarea contextului dinamic.

Henricksen et al folosi mai multe tipuri de **asociații**, precum și raționamentul bazat pe reguli pentru a lua decizii adecvate [Henricksen and Indulska, 2006]. O reprezentare grafică pe bază de grafuri este prezentată, care poate fi, de asemenea, serializată în fișiere XML. Modelul – numit CML, sau Context Modeling Language – se bazează pe modelul obiect-rol.

Recunoașterea contextului și situației

În AmI, caracteristicile vitale de pro-activitate și de anticipare se bazează pe capacitatea sistemului de a recunoaște contextul – sau situația – utilizatorului și să acționeze în consecință. Există două aspecte ale recunoașterii: pe de o parte, recunoașterea de șabloane și de acțiuni, fără un model formal al contextului, efectuată direct de către entitățile care percep contextul; pe de altă parte, recunoașterea situației, bazat pe o reprezentare a informațiilor de context, iar producerea de informații contextuale și recunoașterea situației pot fi efectuate de către entități separate.

2.3 Emergență și sisteme cu auto-organizare

Ori de câte ori un număr mare de entități interacționează intens și formează bucle de feedback, ele formează un sistem complex. Desigur, o implementare ideală de inteligență ambiantă ar putea fi considerată ca un asemenea sistem, conținând un număr mare de dispozitive ce schimbă continuu informații și se adaptează la mediu.

Sistemele complexe sunt greu de controlat, deoarece sunt greu de prezis rezultatele acțiunilor, cum acestea sunt modelate de buclele de feedback complicate din sistem. Sistemele complexe sunt, de asemenea, caracterizate prin proprietăți și fenomene emergente, care sunt neașteptate în raport cu proiectarea elementelor individuale.

Cu toate acestea, proiectarea corespunzătoare a entităților și reglarea proprietăților lor poate duce de fapt la rezultatele scontate la nivelul întregului sistem, și pot fi folosite pentru a controla sistemul, nu prin intermediul unor componente centralizate

sau ierarhice, ci prin intermediul chiar a buclelor de feedback care le produc apariția. Sistemele complexe rezultate au, de asemenea, caracteristici de toleranță la erori și robustețe la nivel de sistem, precum și caracteristici de adaptare.

Utilizarea de agenți reactivi

În domeniul sistemelor multi-agent, cele mai multe exemple care demonstrează proprietăți emergente folosesc agenți reactivi [Serugendo et al., 2006]. Acest lucru se datorează faptului că acestea sunt mai ușor de implementat și de controlat, și acestea sunt potrivite pentru dispozitive mici, cu putere de calcul redusă. Dar, mai mult decât orice, sistemele de agenți reactivi sunt mai ușor de anticipat și de proiectat astfel încât acestea să manifeste proprietăți de auto-organizare sau alți emergenți. Exemple notabile de emergenți în sistemele de agenți reactivi se referă la formarea de anumite structuri geometrice sau comportamentale.

Avantajele caracteristicilor cognitive

Deși sistemele de agenți reactivi pot fi foarte utile, există multe avantaje pe care un agent cognitiv le are față de un agent reactiv. În primul rând, este proactiv. Chiar dacă nu există semnale, percepții sau stimuli din mediu, un agent cognitiv poate acționa de la sine și lua măsuri în conformitate cu obiectivele sale. În al doilea rând, un agent cognitiv este conștient de situația sa și poate raționa despre ea. Agentul cognitiv poate folosi experiența și informațiile despre mediul înconjurător, precum și consecințele acțiunilor trecute pentru a dezvolta un plan mai bun de fiecare dată când apare o situație similară.

Chapter 3

Comportamentul agenților: folosirea auto-organizării

În construirea unui sistem multi-agent pentru Inteligența Ambientă, primul aspect asupra căruia ne-am concentrat a fost comportamentul agenților. Mai precis, modul în care ar trebui să facă schimb de informații agenții, astfel încât informațiile interesante să ajungă la agenții interesați, fără un control centralizat.

Vedem un mediu inteligentă ambientă ca un număr mare de dispozitive care servesc nevoilor utilizatorilor respectivi. Dispozitivele vor gestiona majoritatea timpului diverse informații: furnizarea de informații relevante pentru utilizatori interesați, agregarea, filtrarea și raționamentul despre informații – fiind ”transportoare de informații”. Întrebarea pusă este: dată fiind o anumită informație, cum se va livra acea informație către agenții interesați – agenții care pot folosi această informație pentru a ajuta utilizatorii? Aceasta este o problemă care se adresează nivelului aplicație al unui mediu AmI.

3.1 Abordare

Design-ul sistemului multi-agent – MAS – care este prezentat în acest capitol a pornit de la ideea următoare: la scară realistă, un sistem de AmI va avea de a face cu un număr foarte mare de utilizatori și un număr și mai mare de dispozitive care comunică între ele. Sistemul a fost conceput ținând seama de obiectivele prezentate în capitolul 1 (în special secțiunea 1.5): utilizarea de agenți software cognitivi și a dependenței de context, descentralizare, comportament local, posibilitatea de execuție pe dispozitive cu capacități diferite. Obținerea unui rezultat coerent la nivel global (la nivelul de ansamblu al sistemului de agenți) se face prin utilizarea mecanismelor de auto-organizare.

3.2 AmIciTy:Mi

Nivelul aplicație al AmI poate fi, de asemenea, văzut ca un middleware: asigurarea schimbului dependent de context de informații, folosind hardware-ul și rețeaua pentru a transmite informații, precum și furnizarea de informații relevante pentru componentele specifice aplicației și pentru interfață. Am numit acest middleware AmIciTy:Mi, ca parte a mediului AmIciTy de inteligență ambientă care este în curs de construcție. AmIciTy:Mi impelmentat împreună Gratie Cristian.

Din punctul de vedere al dispozitivelor, middleware-ul este accesibilă numai prin intermediul agenților care execută pe dispozitiv. Interfața dispozitivului comunică cu agentul / agenții software de pe dispozitiv, printr-o structură uniformă de date. Când agentul primește informații noi din exterior sau din dispozitiv, acesta decide cu privire la aceste informații și, în cazul în care consideră că este necesar sau adecvat, trimite informațiile către alți agenți din vecinătatea sa.

Măsuri de context

Deoarece AmIciTy:Mi este concentrat asupra comportamentului agent și asupra controlului schimbului de informații în cadrul sistemului, reprezentarea contextului a fost păstrată simplă. Obiectivul a fost să fie în măsură să controleze diferite aspecte ale fluxului de informații din sistem prin intermediul unor măsuri de context numerice, simple și generice.

Patru măsuri de context au fost folosite, una dintre ele fiind implicită și celelalte trei fiind reprezentate prin intermediul unor valori numerice simple. În primul rând, spațiul este în mod inerent considerat, din cauza structurii sistemului, care se bazează pe comportament și comunicare locale. În al doilea rând, contextul temporal este implementat ca o perioadă de valabilitate pentru fiecare informație. În al treilea rând, fiecare informație se referă la anumite domenii de interes. În sfârșit, fiecare informație poartă o indicație directă de relevanță.

Compatibilitate contextului, sau *relevanța* pentru noile informații se calculează ca o funcție a măsurilor de context asociate cu noua informație și a contextului agentului, compus dintr-o indicație de specialitate. Pentru a putea compara și agrega măsuri diferite, toate sunt în intervalul $[0, 1]$:

- localitatea (proprietatea de a fi local) are o cunaticare explicită ca distanța până la sursa evenimentului: $Dist \in [0, 1]$, cu 0 referind-se la *acest* agent, și crescând asimptotic către 1 pentru distanțe mai mari;
- persistență: $Pers \in [0, 1]$, cu 1 pentru informații valide pentru totdeauna, și 0 pentru informații expirate;
- specialitatea este un vector $Spec \in [0, 1] \times \dots \times [0, 1]$ în care fiecare componentă arată gradul de apartenență la un anumit domeniu de interes, și $\|Spec\| \leq 1$;
- presiunea este tot o valoare $Pres \in [0, 1]$.

În calculul relevanței informațiilor noi, distanța, persistența și presiunea sunt in-

agent are o căsuță de mesaje, o bază de cunoștințe (KB), o listă de obiective disponibile și o listă a planurilor actuale. Informațiile din baza de cunoștințe a agentului sunt stocate ca textit Fapte, care sunt tupluri de forma:

$\langle Agent, knows, Fact \rangle$

Această structură permite agentului să dețină informații cu privire la ceea ce știe, dar, de asemenea, despre ceea ce știu alți agenți. Acesta este modul în care un agent poate calcula specialitatea agenților vecini.

Comportamentul agentului

La începutul fiecărui ciclu agentul verifică mesajele din căsuța de mesaje, prin integrarea faptelor în baza de cunoștințe, dacă acestea sunt noi. Agentul deduce de asemenea că expeditorul cunoaște faptul, cea ce contribuie la cunoștințele agentului cu privire la vecinii săi.

În faza următoare agentul face o listă de scopuri potențiale. Există două tipuri de scopuri pe care un agent poate avea: *Informarea* altor agenți cu privire la unele informații sau *Free* – eliberarea unei părți din capacitatea de stocare. Fiecărui scop îi este atribuită o importanță, și cel mai important obiectiv va fi ales ca intenție. Importanța scopurilor *Inform* este calculată în conformitate cu măsurile de context pentru faptele corespunzătoare: presiune, similitudine între specialitatea faptului și specialitatea agentului, și profunzimea recursivă a faptului (fapte care se referă la agenți mai depărtați sunt mai puțin importante). Importanța pentru scopul *Free* se calculează în funcție de cât de plină este capacitatea de stocare a agentului, atingând o valoare de 1 (cea mai mare importanță), în cazul în care baza de cunoștințe consumă întreaga capacitate disponibilă. Agentul trebuie să aibă întotdeauna o anumită capacitate liberă pentru fapte noi, care provin de la alți agenți.

După alegerea unui scop, agentul face un plan pentru acesta. Pentru *Free*, agentul decide ce fapte să se șteargă. Pentru *Inform*, agentul decide ce vecini să informeze cu privire la faptul corespunzător. Numărul de vecini informați este direct legat de presiunea faptului. Agenții sunt aleși în funcție de specialitatea lor estimată, calculată ca medie a specialității faptelor pe care agentul știe că vecinul le are.

3.3 Evaluare

Implementarea proof-of-concept AmIciTy:Mi s-a axat pe obținerea unor distribuții de informație emergente și coerente, în cadrul sistemului multi-agent, păstrând în același timp o cantitate foarte scăzută de cunoștințe în agenții individuali.

Prototipul a fost implementat în Java, cu suport pentru plasarea agenților într-o structură de tip grilă sau la poziții aleatoare, cu comunicare directă numai între agenți învecinați – la o anumită distanță pentru agenții din poziții aleatoare.

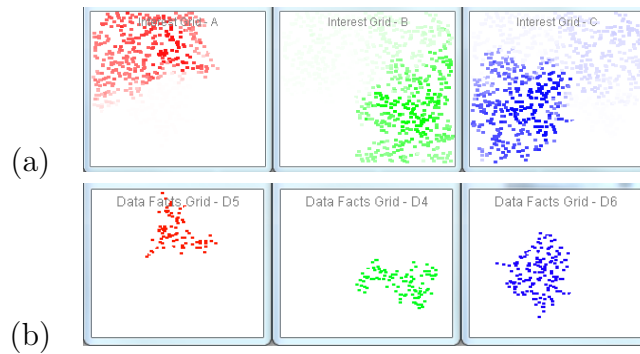


Figure 3.2: (a) Zonele de specializare conform cu cele trei domenii de interes, cu agenți plasați aleator. (b) Distribuțiile de date rezultante.

Experimente cu până la 950 sau 1000 de agenți au fost realizate, axate pe observarea caracteristicilor de răspândire a datelor în sistemul de agenți, cum ar fi: viteza de răspândire, gradul de acoperire atins de fiecare informație individuală, precum zonele preferate de răspândire. Rezultatele experimentelor arată modul în care aceste caracteristici sunt legate de măsurile de contextul asociate cu informațiile.

În scopul de a permite specificarea mai ușoară de scenarii, am folosit fișiere XML pentru a caracteriza scenariile într-un mod simplu și eficient. Fișierele XML utilizează etichete speciale concepute pentru a permite specificarea de scenarii complexe.

Scenariile care au fost studiate folosesc modelul următor: se introduc mai multe informații în sistem, cu specialități diferite, și se lasă să se răspândească până ce acoperirea maximă este atinsă și anumite domenii de interes se stabilesc în sistem; se introduc o serie de informații de "test", de diferite specialități, și se observă modul în care acestea s-au răspândit în funcție de zonele de interes stabilite în prealabil, în timp ce interesul fiecărui agent suferă doar mici modificări; se introduc una sau două informații fără specialitate specială, dar cu presiune foarte mare, și se observă dacă și cat de repede ajung la toți agenții din sistem.

Rezultate experimentale

Rezultatele nu sunt limitate la structura de tip grilă a rețelei (care este însă un mod mai convenabil de vizualizare). Experimentele au fost efectuate, de asemenea, asupra agenților plasați aleator. Rezultatele obținute au fost similare în natură, cu observația că a durat mai mult ca informațiile să se răspândească, ca urmare a mai multor puncte în care agenții au fost prea departe pentru a comunica. Rezultatele obținute cu agenți plasați aleator sunt prezentate în figura 3.2.

Rezultatele arată modul în care măsurile generice de context pot fi utilizate, împreună cu un comportament simplu (și rapid) al agenților, în scopul de a obține un comportament dependent de context. Cunoștințele locale și măsurile simple de context au dus la baze de cunoștințe ale agenților cu nu mai mult de 12 fapte rădăcină, printre care adâncimea medie recursivă a fost de 2, ceea ce înseamnă că foarte puțină de memorie a fost folosită

Chapter 4

Structurarea sistemului multi-agent

Urmând implementării și experimentelor cu proiectul AmIciTy:Mi – prezentate în capitolul anterior, a fost clar faptul că o structură mai bună este necesară pentru sistemul de agenți, dincolo de topologia simplă relativă la spațiu din AmIciTy .

Un factor esențial în efortul de a găsi o topologie mai bună a sistemului a fost de lucru cu limbajul de programare orientat CLAIM (Limbaj computațional pentru agenți autonomi, inteligenți și mobili) și platforma SyMPA (Système Multi-Plateformes d’Agents) [Suna and El Fallah Seghrouchni, 2004]. Pe lângă avantajul de a fi orientat-agent, CLAIM este inspirat de calculul ambient. Având ambienții mobili ca baze ale limbajului, agenții CLAIM sunt caracterizați prin mobilitate, și în special mobilitatea ierarhică. Abordarea noastră de a crea o topologie dependentă de context se bazează pe ideea de corespondență (mapare) între aspectele ierarhice din context și ierarhia agenților. Această abordare a fost validată prin proiectul Ao Dai.

4.1 Abordare

Din punct de vedere al nivelului aplicație, dependența de context este soluția pentru un flux previzibil, natural de informații. Ca și în rețelele sociale și site-uri de cumpărături, se poate presupune că utilizatorul va fi interesat de lucruri care sunt legate de ceea ce el sau ea știe deja, de ceea ce el sau ea face, și de oamenii pe care el sau ea îi cunoaște. Este puțin probabil că cineva sa fie în mod normal interesat de ceva ce nu are nicio legătură cu vreo parte din viața lui sau a ei. Informațiile relevante sunt informații care se referă la contextul în care se află utilizatorul. Contextul are mai multe aspecte: spațiul fizic, timp, activitate (curentă, trecută sau planificată), relații sociale și resurse computaționale. Având în vedere un spațiu în care dimensiunile se referă la aceste cinci aspecte, relevanța informațiilor pot fi

definită ca *proximitate* în acest spațiu.

Cum contextul se bazează pe acest tip de localitate, acest lucru rezolvă de asemenea problema supraîncărcării informaționale. Utilizatorul poate face doar un singur lucru la un moment dat, să fie într-un singur loc, doar un număr de acțiuni din trecut sunt în continuare relevante și numai un număr limitat de acțiuni pot fi planificate. Deci, spațiul contextual al utilizatorului este limitat și va fi legat doar de o cantitate limitată de informații, care se pot fi sortată în funcție de gradul său de relevanță față de contextul actual.

4.2 The Ao Dai Project

Scopul proiectului Ao Dai este de a implementa un scenariu simplu de AmI folosind un sistem multi-agent, în care agenții sunt atribuiți diferitelor elemente din context – locuri, dispozitive, servicii, utilizatorii – și relațiile ierarhice dintre agenți reflectă structura ierarhică a contextului. În scenariu, un utilizator este asistat în mod pro-activ în navigarea pe etajul unei clădiri și în localizarea resurselor computaționale necesare pentru o prezentare.

Proiectul Ao Dai a fost implementat în colaborare cu Thi Nguyen Thuy Nga și Diego Salomone-Bruno, sub supravegherea Prof. Amal El Fallah Seghrouchni. Scenariul prezentat mai jos a fost demonstrat într-un mediu simulat, executând pe două mașini diferite, în cadrul celui de-al 5-lea Wokshopu NII-LIP6, care a avut loc în iunie 2010 în Paris, Franța¹.

Scenariu

În cadrul proiectului, am studiat mai multe scenarii, inclusiv următorul: un utilizator are o întâlnire într-o clădire pe care el / ea nu o cunoaște. La sosirea la etajul potrivit, PDA-ul utilizatorului se conectează automat la un punct de acces fără fir local. Un agent CLAIM execută pe PDA – vom numi acest agent *PDA*. Un alt agent execută pe o mașină locală și gestionează contextul etajului – numim acest agent *Floor*. *Floor* detectează prezența utilizatorului, și instruește agentul *PDA* să se deplaseze în structura sistemului de agenți pentru a deveni fiu al *Floor*. *Floor* creează, de asemenea, un agent nou – numit *Navigator* – și-l instruește să se deplaseze ca fiu al lui *PDA*. *Navigator* este un agent care oferă utilizatorului servicii de navigație care sunt disponibile și au sens numai în contextul etajului respectiv. *PDA* poate detecta că ecranul său este prea mic pentru a afișa harta în mod corespunzător, astfel că *PDA* va iniția în mod pro-activ căutarea unui ecran mai mare, în zona din apropiere. *PDA* trimite interogarea către părintele său – *Floor* – care, la rândul său, localizează în rândul fiilor săi un agent *Screen*, care gestionează un ecran fizic care se

¹Workshop internațional organizat în colaborare de către Institutul Național de Informatică din Tokyo și Laboratorul de Informatică de la Universitatea Paris 6. Detalii la <http://www-desir.lip6.fr/~herpsonc/5workshopNii/program.htm>

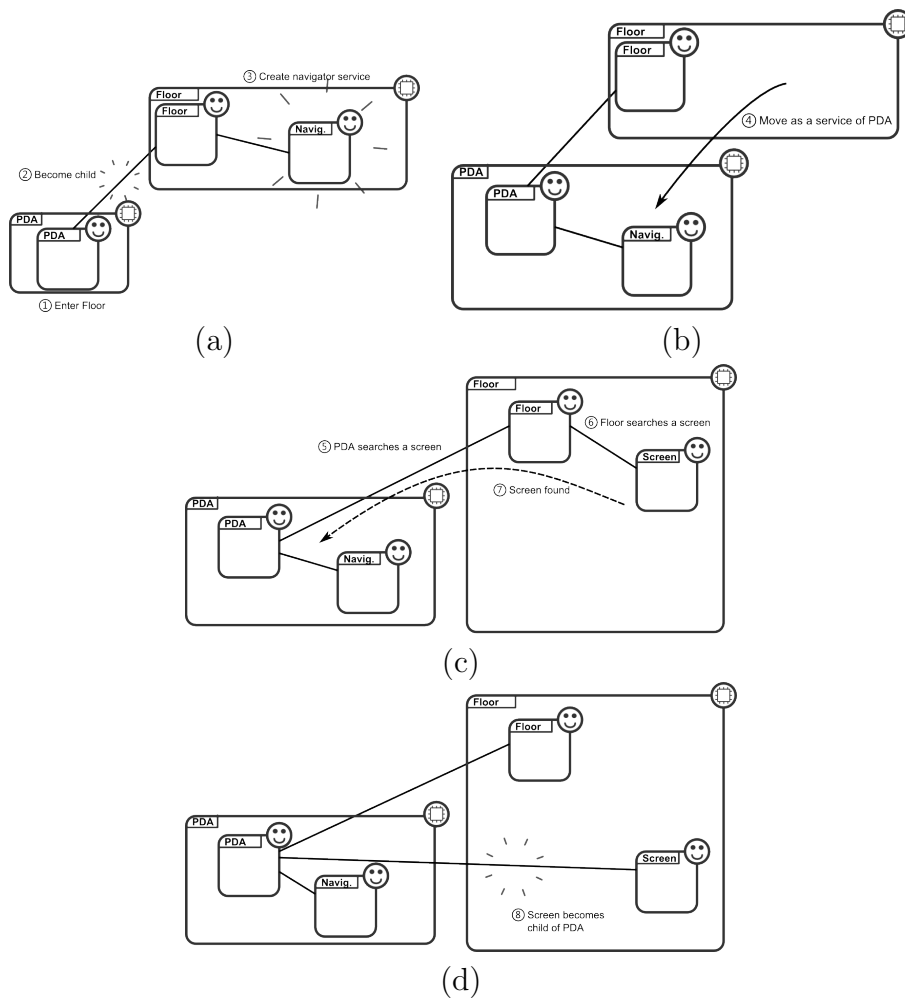


Figure 4.1: Pașii scenariului Ao Dai: un utilizator cu un PDA intră pe etajul clădirii, *Floor* acceptă pe *PDA* ca fiu și creează serviciul *Navigator* (a); *Navigator* este trimis ca fiu al lui *PDA* (b); *PDA* necesită un ecran, care este găsit ca fiu al lui *Floor* (c); *PDA* capătă controlul asupra *Screen*.

potrivește cerințelor, este situat în apropiere de pentru utilizator și este disponibil. *Screen* răspunde interogării și *PDA* îi solicită să se mute pentru a deveni fiul său. Fiind fiu al lui *PDA* marchează, de asemenea, faptul că *Screen* este utilizat de către utilizator, și *PDA* are controlul asupra informațiilor afișate.

Implementare

În implementarea scenariului, există trei tipuri principale de agenți: agentul *Site* (cu subtipurile *Floor* și *Office*), agentul *Device / Service* (de exemplu, agentul *Navigator*, *Agenda*, *Screen*) și agentul *PDA*, acesta din urmă cu rolul specific de a reprezenta utilizatorul în timpul simulării.

Pentru a păstra ierarhia de agenți, agenții pot interacționa doar cu părintele și fiii lor. Avantajul de a folosi un astfel de protocol împreună cu maparea contextului peste

ierarhia agenților este că interogările se vor termina, de obicei, repede, presupunând că utilizatorul, de cele mai multe ori, va necesita dispozitive care este probabil să existe în contextul său. Căutarea este executată întâi în contextul actual, apoi în contextul părinte și în contexte învecinate

Chapter 5

Îmbunătățirea dependenței de context

Există multe definiții pentru ceea ce este contextul. În domeniul Inteligenței Ambientale și al aplicațiilor dependente de context, există o definiție care este citată mai mult decât altele, dată de Dey în 2001: „Contextul este orice informație care poate fi folosită pentru a caracteriza situația unei entități. O entitate este o persoană, loc, sau un obiect care este considerat relevant pentru interacțiunea dintre utilizator și aplicație, inclusiv utilizatorul și aplicația înșiși” [Dey, 2001]. În ceea ce privește *dependența de context*, este proprietatea unei aplicații de a se adapta automat la contextul perceput, prin schimbarea comportamentului aplicației în consecință [Chen and Kotz, 2000].

Pentru a exemplifica, să reiterăm exemplul nostru scurt, dintr-o secțiune anterioară: un cercetător în ultima zi înainte de termenul limită pentru o conferință importantă, scriind articolul (care a fost lăsat pentru ultimul moment) ar putea fi deranjat de notificarea despre un mesaj de la un coleg cu privire la o legătură interesantă care se referă la domeniul lor de cercetare. În mod normal, asta nu s-ar întâmpla, dar în *acest context special*, este mai bine a afișa notificarea pentru mesaj numai în ziua următoare.

O aplicație dependentă de context (și o aplicație AmI în special) trebuie să poată **accesa informația de context** (de exemplu, să știe despre activitatea obișnuită a cercetătorului, să știe când este termenul limită pentru articol), trebuie să **înțeleagă contextul** (de exemplu, să înțeleagă relațiile dintre diverse fapte din context, să evalueze importanța mesajului primit) și trebuie să fie în măsură să **decide cu privire la acțiunea corectă din punct de vedere al contextului** (de exemplu, să știe despre experiența anterioară a utilizatorului și despre așteptările sale, și, de asemenea, să poată gestiona notificările).

Scopul nostru este de a **integra dependența de context într-un sistem multi-agent pentru inteligență ambiantă**, astfel ca agenții să acceseze și să

schimbe în mod natural informații de context.

Acest capitol prezintă o abordare pentru reprezentarea contextului și pentru comportamentul dependent de context, în contextul implementării unui middleware pentru AmI ca cel prezentat în capitolul 3, dar mai bun. După o perspectivă holistică asupra dependenței de context, se dau detalii cu privire la cele două aspecte ale dependenței de context pe care le integrează: în primul rând, o mai bună reprezentare pentru informațiile de context, precum și mijloace de a recunoaște informațiile relevante în având în vedere contextul, folosind *grafuri de context*, *șabloane de context* și *potrivirea contextului*; în al doilea rând, o topologie orientată către context pentru sistemul de agenți, care se bazează pe conceptele dezvoltate în capitolul 4, dar folosind o mapare mai generală între context și topologia sistemului, care este de asemenea integrată cu grafurile de context din interiorul agent. În sfârșit, este descris un comportament îmbunătățit al agentului, pe baza elementelor de dependență de context din secțiunile anterioare.

5.1 O abordare holistică pentru dependența de context

Abordarea folosită pentru construirea unui sistem multi-agent pentru nivelul aplicație al inteligenței ambiante are două aspecte: topologia dependentă de context a sistemului de agenți și utilizarea de șabloane de context, pentru a recunoaște informațiile relevante. Pe lângă aceste două elemente, comportamentul agenților se bazează pe schimbul de informații cu agenți vecini, în scopul de a răspândi informațiile interesante (cu privire la contextul actual).

Un model formal pentru sistemul multi-agent

Sistemul multi-agent este organizat pe trei niveluri: containere, agenți, și cunoștințe / informații de context. Starea actuală a sistemului reprezintă starea actuală a lumii, în conformitate cu punctul de vedere al sistemului ca întreg. De asemenea, în grafurile de context deținute de fiecare agent reprezintă starea actuală a lumii din perspectiva agentului.

Containerele sunt atribuite mașinilor fizice, și la un moment de timp fiecare agent execută pe un singur container. Fiecare agent are cunoștințe, reprezentate prin graful său de context, precum și de șabloanele sale de context. Fiecare dintre cele trei niveluri formează un graf, care este un subgraf al întregului graf de trei niveluri, pe care îl vom numi **Tri-Graph**.

Containerele formează un subgraf complet

$$ContainerGraph = (Containers, Connections)$$

Agenții formează subgraful *AgentGraph* în care muchiile sunt etichetate cu tipuri

de relații relativ la contextul partajat (vezi Secțiunea 5.3): *is-in*, *part-of*, etc.:

$$AgentGraph = (Agents, AgentRelations);$$

$$Agents \subset AgentNames;$$

$$AgentRelations = \{(A_i, A_j, Relation)\} \text{ unde } A_i, A_j \in Agents$$

și $Relation \in AgentRelationTypes$;

$$AgentRelationTypes = \{is-in, part-of, of, in, controlled-by, executes-on\}.$$

Asocierea între **agenți și containere** este făcută printr-o componentă suplimentară – $AgentLocations \subset Agents \times Containers \times \{resides-on\}$, care leagă cele două niveluri.

Un **agent** A este un tuplu $A(Name, CG_A, Patterns, \mathcal{R}, I, Goallist)$, care conține numele agentului, graful de context, setul de șabloane, relațiile cu alți agenți, informații despre interesul altor agenți, și lista de scopuri.

Tri-Graph este format prin reuniunea grafurilor de containere, agenți și cunoștințe:

$$Tri-Graph = (Nodes, Edges), \text{ unde}$$

$$Nodes = Containers \cup Agents \cup \bigcup_{A \text{ agent}} CG_A.Concepts$$

$$Edges = Connections \cup AgentRelations \cup AgentLocations$$

$$\cup \bigcup_{A \text{ agent}} CG_A.Relations$$

5.2 Dependență de context în interiorul agentului

Fiecare agent A are un *Context Graph* $CG_A = (V, E)$ care conține informațiile relevante în raport cu funcția sa:

$$CG_A = (V, E), \text{ where } V \subset Concepts \text{ and } E = \{edge(from, to, value, persistence) \mid , from, to \in Concepts, value \in Relations, persistence \in (0, 1] \}$$

Elementele din *Concepts* și *Relations* sunt șiruri de caractere sau URI; *Relations* conține de asemenea șirul vid.

Pentru ca agenții să poată recunoaște situația, introducem șabloanele de context. Un șablon reprezintă un set de asocieri care a fost observat că apare de mai multe ori și poate apărea din nou. Un agent are un set *Patterns*:

$$Patterns = \{(G_s^P, relevance, persistence) \mid s \in PatternNames, G_s^P \text{ un graf-șablon, } relevance, persistence \in (0, 1]\}.$$

Un **șablon de context** s este definit ca un graf $G_s^P = (V_s^P, E_s^P)$.

$$V_s^P = \{v_i^P(label) \mid label \in Concepts \cup \{?\}\}$$

$$E_s^P = \{(from, to, label, characteristic, actionable)\},$$

$$\text{cu } from, to \in V_s^P, label \in Regexprs(Relations),$$

$$characteristic, actionable \in (0, 1]$$

Trăsătura *characteristic* definește cât de caracteristică este muchia pentru șablon,

și influențează măsurarea a cât de bine un șablon se potrivește cu un subgraf; *actionability* măsoară cât de corect ar fi dacă agentul ar deduce existența acestei muchii în graful de contextul dacă șablonul corespunde unui subgraf în CG , dar această muchie lipsește. Șablonul este, de asemenea, caracterizat prin două alte proprietăți: *relevance* arată cât de importantă este o informație din graful de context, în cazul în care se potrivește peste acest șablon; *persistence* arată pentru cât timp o informație nouă va persista după ce a fost potrivită cu acest șablon.

Recunoașterea contextului prin potrivire

Un agent are un set de șabloane pe care le potrivește peste contextul actual, și peste informațiile pe care le primește de la alți agenți.

O *potrivire* i între un șablon de context G_s^P și graful de context CG_A al unui agent A este definită ca $M_{A-si}(G'_A, G_m^P, G_x^P, f, k_f)$. Mai precis, $G'_A \subset CG_A$ este o potrivire perfectă pentru *partea rezolvată* G_m^P a șablonului G_s^P . Ceea ce rămâne este *partea nerezolvată* G_x^P (numită și *problemă*). Părțile rezolvată și nerezolvată ale șablonului nu se intersectează.

Funcția $f : V_m^P \rightarrow V'$ este injectivă, și:

$$(1) \forall v_i^P \in V_m^P, v_i^P.label = ? \text{ or } v_i^P.label = f(v_i^P).label$$

și

(2) $\forall e_i^P \in E_m^P$, with $e_i^P.from, e_i^P.to \in V_m^P$, $e_i^P.value$ se potrivește (ca expresie regulată) cu secvența $value_0 \cdot value_1 \cdot \dots \cdot value_p$ de valori ale muchiilor $e_0 = (f(e_i^P.from), v_{a_0}, value_0) \dots e_k = (v_{a_{k-1}}, v_{a_k}, value_{k+1}) \dots e_p = (v_{a_{p-1}}, f(e_i^P.to), value_p)$, cu $k = \overline{1, p-1}$, $a_k \in \{0, \dots, |V'| - 1\}$ și $v_{a_k} \notin f(V_m^P)$.

Mai precis, orice vârf non-? din partea rezolvată corespunde unui vârf diferit din G'_A ; fiecare muchie care nu este expresie regulată din partea rezolvată corespunde unei muchii din G'_A ; și fiecare muchie expresie regulată din partea rezolvată corespunde unei serii de muchii din G'_A . O condiție suplimentară este ca G'_A să nu conțină alte noduri și muchii (G'_A este minim).

Numărul $k_f \in (0, 1]$ indică cât de bine șablonul G_s^P se potrivește peste G'_A în potrivirea M_{A-si} , și este dat ca suma normalizată a factorilor *characteristic* al muchiilor potrivite, adică:

$$k_f = \frac{\sum_{e_i^P \in E_m^P} e_i^P.characteristic}{\sum_{e_j^P \in E_s^P} e_j^P.characteristic}$$

Un **algoritm pentru potrivirea contextului** a fost dezvoltat, bazat pe extinderea potrivirilor pornind de la potriviri de o singură muchie.

5.3 Dependența de context în exteriorul agentului

Nevoia de structurare a sistemului multi-agent într-o manieră conștientă de context apare pentru că comportamentul agentului – dezvoltat în AmIciTy:Mi (vezi capitolul 3) – se bazează pe interacțiunea locală și pe cunoștințe locale. Dar, ”local” nu trebuie neapărat să însemne local în spațiu. Alte aspecte ale contextului pot fi utilizate. Acestea au fost explorate pentru prima oară în proiectul Ao Dai – a se vedea capitolul 4 – în care sistemul de agenți a fost ierarhizat în scopul de a reflecta natura ierarhică a două tipuri de context – spațial și computațional.

Sistemul de agenți se bazează pe schimbul dependent de context de informații între un agent și vecinii săi. Dar un agent trebuie să facă schimb de informații doar cu un alt agent care ar putea considera informațiile ca relevante. Și asta se poate întâmpla numai în cazul în care agenții au în comun unele tipuri de context. De exemplu, în cazul în care agenții fac parte din aceeași activitate, sau în cazul în care utilizatorii lor sunt situați în același spațiu. Doi agenți care nu împărtășesc niciun context, nu ar avea nici o informație care să fie relevantă pentru ambii. Prin urmare, **topologia sistemului este indusă de context**: dacă doi agenți împărtășesc context, aceștia trebuie să fie vecini.

Ca și în proiectul Ao Dai, vom păstra diferite tipuri de agent pentru fiecare aspect al contextului. Mai precis, un agent va fi interesat în primul rând de informații cu privire la un anumit aspect de context, de exemplu, referitoare la un loc, la un utilizator, etc.

Am definit următoarele tipuri de agenți și de relații de vecinătate, pentru a se potrivi diferitelor aspecte ale contextului:

- pentru contextul spațial – agentul *Place* și relația *is-in*;
- pentru contextul computațional – agenții *Device* și *Service* și relațiile *of* (către agentul de care este legat serviciul) și *executes-on* pentru servicii și *controlled-by* (către utilizatorul care îl utilizează) pentru dispozitive;
- pentru contextul de activitate – agentul *Activity* și relația *part-of*;
- pentru contextul social – agenții *User* și *Group* și relațiile *in* (care leagă un utilizator la un grup) și *connected-to* (care leagă doi utilizatori).

În timp ce contextul temporal este un aspect de context pe care îl considerăm, nu există un agent specializat pentru intervale de timp (care ar fi elementul ierarhic de context temporal), pentru că un agent care gestionează un interval de timp nu are sens, din moment ce reprezentarea contextului intern, precum și relațiile dintre agenții, se întâmplă în prezent – contextul temporal este prin urmare permanent împărtășit. Cu toate acestea, dacă este necesară specificarea de indicații de intervale de timp – de exemplu, în cazul activităților, disponibilitatea de dispozitive sau de servicii, și grupurile temporare de utilizatori de – aceste informații pot fi atașate la cunoștințele agentului.

5.4 Un comportament îmbunătățit pentru agenți

Deși comportamentul agenților AmIciTy prezentat în secțiunea 3.2 asigură o diseminare coerentă de informații în sistemul de agent, controlată de măsuri de context, atât topologia sistemului cât și măsurile de context care au fost utilizate au fost destul de simple. În secțiunile precedente ale acestui capitol, am formalizat o manieră mai bună de manipulare a informațiilor contextuale, precum și o topologie îmbunătățită pentru sistemul de agenți.

Principiile pe care comportamentul îmbunătățit se bazează sunt cele care au fost prezentate pe parcursul acestei lucrări: un agent ar trebui să trimită informații de care este interesat vecinilor pe care agentul crede că ar putea fi interesați de ele; topologia sistemului este indusă prin context: dacă doi agenți împărtășesc context, aceștia ar trebui să fie vecini.

Toți agenții sunt definiți de trei comportamente esențiale: **potrivirea de șabloane** (recunoașterea situației și pro-activitatea în raport cu utilizatorul), **schimbul de informații** (pro-activitate referitoare la alți agenți), și **integrarea informațiilor** (reactivitate). Comportamentul agentului este aproximativ identic cu comportamentul prezentate în secțiunea 3.2, doar că, în loc de a lucra cu specialități și presiuni, acum agentul lucrează cu modele de context și potrivirea contextului.

Toate acțiunile pe care un agent le poate executa sunt legate de potrivirile de context și de muchiile adăugate (crearea unor relații între concepte). Din punctul de vedere al sistemului multi-agent, un caz special de acțiune este crearea unor relații între agenții. Potrivit șabloanelor sale, un agent poate deduce că, în unele cazuri, o relație ar trebui să existe cu un alt agent, prin adăugarea unei muchii în CG_A între conceptele care reprezintă pe sine și celălalt agent. Dacă este cazul, agentul va informa, de asemenea, gestiunea sistemului multi-agent (precum și alte agenți) de relația nouă. Același lucru se întâmplă atunci când o muchie dispare.

Un exemplu extins

În scopul de a da o imagine mai clară despre modul în care funcționează comportamentul dependent de context pentru, să ne întoarcem la scenariul de referință. Scenariul descrie următoarea situație: Alice, un utilizator de servicii de AmI, este într-un tren, pe drumul spre cursul de informatică de la Universitatea la care este studentă. Trenul lui Alice va avea câteva minute întârziere. Totodată, profesorul care va preda cursul așteaptă începerea cursului, folosind serviciul *CourseStarter*, și urmează să fie notificat despre studenții vor întârzia, și pentru cât timp.

Considerând un agent alocat pentru fiecare element de context, topologia sistemului este prezentată în figura 5.1. Cu această topologie, atunci când informația că Alice va întârzia este generată de *Scheduler* (folosind informații trimise către Alice de *Train*), acesta va informa agentul care gestionează participarea ei la Curs, care va difuza aceste informații către *CSCourse*, care va informa *CourseStarter*. În acest fel,

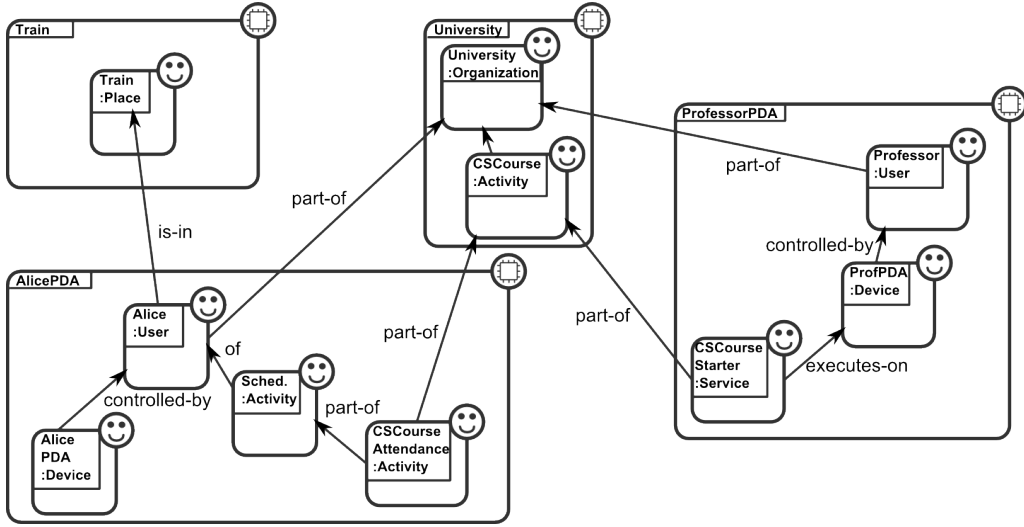


Figure 5.1: Topologia agenților pentru scenariul de referință. Agenții care se execută pe aceeași mașină sunt încercuși corespunzător.

se răspândesc informații între agenții având un context comun, aici contextul comun fiind în principal legat de activitate. Având în vedere comportamentul agentului descris, pot exista și alți agenți care primesc aceste informații, dar ei vor renunța la ele, cum informațiile nu sunt relevante pentru ei. Acest lucru este decis de șabloanele de context. Putem presupune că agentul *Train* știe că, printre altele, utilizatorul Alice este în tren, având deci $Alice \xrightarrow{is-in} Train \#1691$ ca subgraf al CG_{Train} . Este rezonabil să copnsiderăm că agentul *Train* are un șablon $TrainRide(\xrightarrow{on} Train \#1691) (\xrightarrow{to} ?Place) (\xrightarrow{within} ?Time Interval) (\xrightarrow{part-of | of})* ?User \xrightarrow{is-in} Train \#1691)$. Acest șablon se potrivește cu informațiile despre intervalul de timp ale unei călătorii cu trenul care este parte din activitățile unui utilizator (și are o destinație specifică). Informații și algoritmi specifice domeniului vor permite trenului să estimeze intervalul de timp pentru destinațiile tuturor utilizatorilor. Apoi, aceste informații vor fi trimise utilizatorilor care sunt potențial interesați – mai precis, fiecare potrivire va fi trimisă utilizatorului corespunzătoare.

Chapter 6

O nouă platformă pentru aplicații AmI

Pentru a putea testa modelul bazat pe agenți pe care le-am dezvoltat, o implementare este necesară. Deși unele părți ale acestei lucrări au fost testate folosind implementări mai simple sau parțiale a sistemului, rezultatul final ar trebui, pe de o parte, să integreze toate conceptele teoretice pe care le-am prezentat și, pe de altă parte, să prezinte funcționalități cum ar fi comunicarea distribuită, mobilitate și vizualizarea centralizată a agenților.

Acest capitol oferă detalii cu privire la implementarea platformei Ao Dai (care este diferită de prototipul Ao Dai). Realizarea platformei a fost un efort colaborativ a lui Andrei Olaru, Nguyen Thi Thuy Nga și Marius-Tudor Benea, sub supravegherea prof. univ. Amal El Fallah Seghrouchni și cu asistența lui Cédric Herpson.

Pentru a face arhitectura platformei de ușor de extins și îmbunătățit, am adoptat o abordare modulară, bazată pe mai multe componente și niveluri, după cum urmează:

- componenta *Core*, ce conține clasele de agenți, organizat pe mai multe niveluri:
 - comunicare, mobilitate, și gestionare pentru agenți – sunt utilizați agenți JADE;
 - trasabilitate și vizualizare – agenții trebuie să transmită date despre activitatea lor, precum și despre vecinii lor, către o entitate centralizat. Fiecare agent de afișează de asemenea, o fereastră pe ecranul mașinii gazdă;
 - mobilitate ierarhică pentru agenții – protocoale și comportamente care să permită agenților să se deplaseze în mod automat, împreună cu părinții lor;
 - acces la servicii web – funcționalitate pentru a expune agenții ca servicii web și pentru a permite agenților accesul la servicii web;
 - interpretarea și executarea S-CLAIM – un parser pentru fișiere S-CLAIM de descriere a agenților, și componentele care transformă definiția în com-

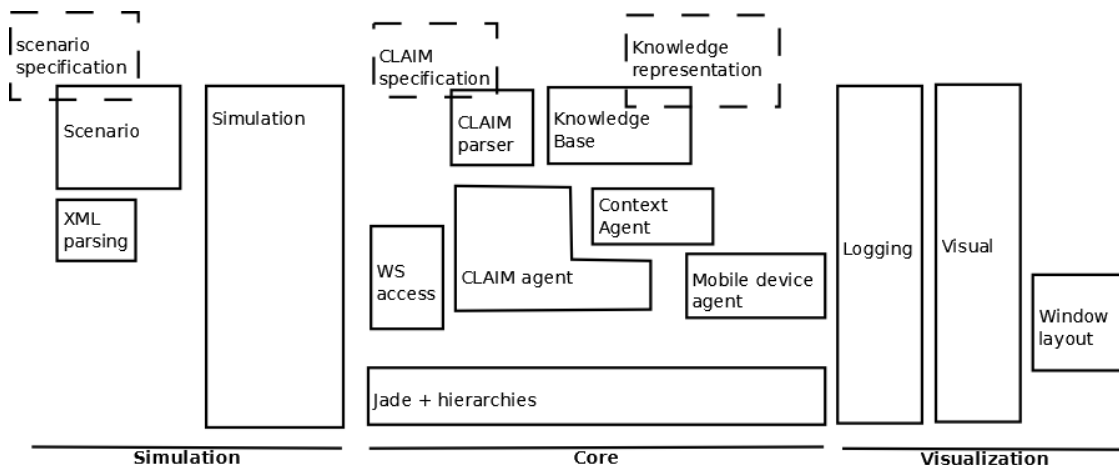


Figure 6.1: O reprezentare vizuală a componentelor platformei Ao Dai. Cu linie continuă, component reale ale implementării. Cu linie punctată, specificații și formate care caracterizează intrările componentelor.

- portamentul real;
- Baza de cunoștințe – o componentă interschimbabilă care permite accesul la cunoștințe prin intermediul unui set standard de funcții;
- dependența de context – utilizare potrivirii contextului pentru rezolvarea problemelor și schimbul de informații de context relevante.
- componenta *Simulation*, care servește pentru executarea de scenarii repetabile:
 - folosește ca date de intrare fișiere XML care definesc complet scenariul de execuție;
 - execută agenții în funcție de scenarii, pe containerele și mașinile specificate;
 - trimite mesaje eveniment ”externe”, echivalente cu percepțiile agenților într-un mediu real;
- componenta *Visualization*, care asigură o vizualizare centralizată a activității agenților:
 - primește rapoartele de jurnal de evenimente (log) și evenimentele de mobilitate de la agenți;
 - afișează toate jurnalele într-un mod centralizat, cronologic;
 - afișează structura sistemului (topologie), arătând relațiile dintre agenții;
 - oferă componente pentru plasarea automată a ferestrelor agent pe ecranul mașinii pe care se executa.

O viziune informală a acestor componente este prezentată în figura 6.1. În următoarele secțiuni vom da detalii cu privire la blocurile din platformă și la limbajul orientat-agent S-CLAIM.

Componente constructive

O prima componentă constructivă a platformei Ao Dai a fost JADE Agent Devel-

opment Framework ¹. Aceasta este implementată în Java. Ea poate fi executată pe mai multe calculatoare, agenții se pot alătura sau pot părăsi în orice moment platforma (sistem deschis), și platforma se ocupă de comunicarea între agenți, mobilitate și managementul agenților. Fiind construită peste Java, JADE este ușor de lansat pe platforme diferite. Module separate (plug-ins) permit execuția de agenți de pe dispozitive mobile (smartphone-uri Android de exemplu) și interoperabilitatea cu serviciile web.

Pentru logare activitatea agentului, am dezvoltat un wrapper pentru Apache log4j ², care permite configurarea rapidă a unui jurnal care prezintă ieșirea la consolă, o afișează într-o fereastră, și o trimite (sub forma unui mesaj JADE) la agentul de vizualizare.

O componentă a fost, de asemenea dezvoltată pentru execuția de agenți S-CLAIM pe dispozitive mobile Android ³, folosind JADE-LEAP ⁴. Acest add-on permite executarea unui container JADE pe dispozitivele care rulează sistemul de operare Android, care să permită executarea de agenți JADE pe dispozitiv, precum și deplasarea agenților spre și dinspre smarphone, văzând practic dispozitivul Android în același mod ca o stație de lucru. Modificări au fost necesare, cu toate acestea, la componenta de vizualizare pe dispozitive Android, din motive de compatibilitate a platformei, dimensiunea ecranului, etc.

Pentru interoperabilitatea agenților Ao Dai cu alte componente și infrastructuri, agenții integrează primitive pentru accesul la servicii web atât SOAP cât și RESTful, și pentru expunerea capabilităților agenților ca servicii web. Acest lucru a fost realizat cu ajutorul add-on-urilor JADE WSIG și WSDC.

Componenta de plasare a ferestrelor – pentru plasarea de diverse ferestre (de exemplu, fiecare agent are propria fereastră) pe ecran – a fost portată din proiectul AmIciTy:Mi. Avantajul de a folosi un instrument de plasare automată a ferestrelor este faptul că, atunci când există multe ferestre, utilizatorul nu are nevoie să mute manual fiecare fereastră în poziția dorită, la fiecare rulare a simulării.

S-CLAIM

Principalul nostru obiectiv în crearea limbajului S-CLAIM ("Smart CLAIM") a fost de a-l face ușor de utilizat, astfel că ar putea fi un instrument excelent chiar și pentru cei care nu au fost foarte familiarizați cu alte limbaje de programare. Acest lucru a fost, de fapt, și scopul limbajului CLAIM (Limbaj computațional pentru agenți autonomi, inteligenți și mobili) – de a fi un limbaj orientat-agent care să fie simplu de folosit de către proiectanții de agenți.

Semantica S-CLAIM este strâns bazată pe semantica CoCoMo dezvoltată de Ab-

¹<http://jade.tilab.com/>

²<http://logging.apache.org/log4j/>

³<http://www.android.com/>

⁴<http://jade.tilab.com/community-addons.php>

delkader Behdenna, în timpul stagiului de Master la LIP6 în 2009. Ea sunt o simplificare a semanticii limbajului CLAIM [Suna and El Fallah Seghrouchni, 2004], care reduce lista de primitive doar cele care sunt caracteristice pentru gestionarea agenților și pentru interacțiune. S-CLAIM specifică primitive pentru următoarele clase de acțiuni: comunicare inter-agent, management al agenților, managementul cunoștințelor și unele primitive de control. Acest lucru lasă la o parte toate părțile algoritmice de descriere a agentului, funcții de procesare, aritmetica sau operație logică, etc. Acestea pot fi implementate în alte limbaje de programare, și pot fi invocat în aceeași manieră ca și primitivele specifice S-CLAIM.

Sintaxa de invocare a primitivelor este de tip LISP. Invocarea este formată din paranteze, numele funcției sau primitivei, și parametrii. În timp ce limbajul nu este tipat, fiecare primitivă S-CLAIM face ipoteze cu privire la tipul parametrilor săi.

Scenarii și vizualizare

Una dintre funcționalitățile esențiale pentru a putea testa în mod corect o platformă ca Ao Dai este posibilitatea de a executa simplu scenarii în mod repetat. Acest lucru înseamnă, pe de o parte, că, odată ce scenariul de test este configurat, executarea scenariului se poate face prin simpla apăsare a unui buton. Pe de altă parte, înseamnă că este posibilă executarea exact a aceluiași scenariu din nou, cu exact aceleași rezultate. Scenariul este specificat prin intermediul unui fișier XML care conține informații cu privire la containere și la agenții care sunt creați, cunoștințele lor inițiale, precum și cu privire la evenimentele generate.

Componenta de vizualizare a platformei oferă trei funcții: de logare centralizată, vizualizare centralizată a sistemului de agent, și un sistem automatizat de plasare a ferestrelor pe ecran. Logarea centralizată și vizualizarea se obțin prin trimiterea, din stratul de vizualizare din fiecare agent, și folosind wrapperul pentru log4j, mesaje de logare a agentului, precum și rapoarte cu privire la deplasarea ierarhică, către un agent specializat *Visualization Agent* care sortează mesajele din jurnal după timestamp-ul lor și le afișează în ordine.

Structura agentului

În platforma Ao Dai, agenții sunt organizați intern pe mai multe niveluri. Acest lucru este util pentru modularitatea sistemului, pentru separarea preocupărilor (concern separation) și pentru ușurința de depanare. Mai mult decât atât, noi niveluri ale agentului pot fi adăugate fără prea mult efort. Nivelurile sunt următoarele:

- JADE GuiAgent – agenți de bază JADE care poate avea o interfață grafică. Acest nivel gestionează deplasarea agenților și comunicarea la nivelul platformei JADE, și este baza a întregii structuri a agenților Ao Dai.
- VisualizableAgent – se ocupă de vizualizarea agentului din două puncte de vedere: pe de o parte, le oferă nivelurilor superioare obiectul *log* care adună datele de logare ale agentului, iar acesta raportează datele de logare și deplasarea / schimbarea de părinte agent agentului de vizualizare; pe de altă

- parte, gestionează fereastra agentului, și o integrează în layout-ul de pe ecran;
- WSAgent (Web Service Agent) – oferă funcționalități utilizate de către agentul S-CLAIM pentru a expune capacitățile sale ca servicii web, prin intermediul JADE WSIG, și, de asemenea, pentru a accesa servicii web SOAP sau RESTful, prin intermediul JADE WSDC;
- HierarchicalAgent – gestionează mobilitatea ierarhice a agenților, expunând obiecte pentru relațiile ierarhice între agenții, precum și implementarea de comportamente atât pentru instruirea fiilor unui agent de a urmări agentul, cât și pentru ca un agent să primească ordine de deplasare de la părinte; setările agentului pot permite agenților să fie legați de containerele lor, acest lucru însemnând ca ei nu își vor urma părintele în deplasare;
- agentul S-CLAIM – agentul care execută utilizând descrierea S-CLAIM; conține tabelele de simboluri ale agentului, și, de asemenea, un set de descrieri de comportament care descriu capacitățile agentului; acest nivel accesează componenta Bază de cunoștințe pentru adăugarea, eliminarea, sau interogarea de cunoștințe.

Platforma Ao Dai a fost în primul rând proiectată pentru testarea și simularea de sisteme multi-agent pentru Inteligența Ambientă. Prin intermediul nivelului *HierarchicalAgent* al agenților Ao Dai, o dată ce părintele agentului este definit, agentul se va deplasa împreună cu părintele său. Deplasarea ierarhică înseamnă că un agent păstrează (o parte din) contextul său, atunci când se mișcă, și se păstrează în contextul părintelui său, atunci când se mută părintele. Modul în care limbajul S-CLAIM este construit duce la o tendință a agenților de a comunica în principal doar cu părintele și fiii lor. Nu în ultimul rând, programarea în S-CLAIM se bazează în mare măsură pe șabloane. În timp ce în implementarea actuală nu sunt șabloane bazate pe grafuri, ci șabloane liniare, construcțiile limbajului sunt orientate spre utilizarea de structuri cu valori incomplet legate.

6.1 Experimente

A fost, de fapt, în scopul testării acestei platforme că scenariul de referință a fost creat, în colaborare cu echipa de la Honiden-Lab de la Institutul NII din Tokyo. Pentru primele teste ale platformei Ao Dai, am folosit doar anumite segmente din acest scenariu.

Un număr mare de experimente au fost efectuate cu platforma Ao Dai, și multe altele vor urma. Platforma a fost testat pe o singură mașină, și apoi într-un mediu distribuit, și folosind, de asemenea, agenți care s-au deplasat către și dinspre un dispozitiv Android, ceea ce înseamnă că platforma este acum chiar mai potrivită pentru dezvoltarea de aplicații AmI, cum smartphone-urile sunt fundamentale în arhitectura și scenariile de AmI.

Dar cel mai important, platforma a fost testat în laboratorul SmartRoom de la

Honiden-Lab. Platforma a fost executată pe mai multe calculatoare, și agenții au comunicat cu componentele SmartRoom prin intermediul serviciilor web.

În timp ce nici unul dintre dezvoltatorii de platforma Ao Dai nu a fost prezent la Honiden-Lab înainte de, sau în timpul simulării, echipa de la Honiden-Lab a instalat cu succes și a experimentat cu platforma, la prima lor încercare, fără incidente semnificative. În plus, membrii echipei au făcut cu succes mici modificări în codul S-CLAIM, fără a avea nici o experiență anterioară sau contact cu limbajul. Platforma a fost prezentată și demonstrată la al 6-lea Workshop internațional NII-LIP6, ținut în Octombrie 2011, în Paris, Franța.

Succesul experimentelor și al demonstrației arată că platforma este ușor de utilizat și de executat. JADE oferă interoperabilitate și multe componente utile și add-on-uri. S-CLAIM oferă un limbaj de programare pentru agenți simpli, ușor de interes, care nu necesită experiență anterioară. Integrarea de servicii web a arătat că platforma pot interoperă cu alte componente și infrastructuri, într-un mod fiabil, cu aproape niciun moment petrecut pentru integrare.

Chapter 7

Concluzii

Întrebarea care aceasta cercetare a răspuns este ”Cum trebuie construit un sistem multi-agent pentru Inteligența Ambientă?”. Cerințele pentru această cercetare au fost, printre altele, distribuția sistemului, utilizarea agenților cognitivi, a utilizării mecanismelor de auto-organizare, și cu siguranță nu în ultimul rând, integrarea dependenței de context.

Ideea centrală care caracterizează rezultatul acestei lucrări este **integrarea contextului într-un MAS pentru AmI, într-un mod care să permită agenților gestionarea și partajarea în mod natural a informațiilor de context**, fiind în același timp capabil de a efectua, sarcini specifice aplicației sau domeniului, și respectând, de asemenea, cerințele care au fost stabilite.

Principalele contribuții originale ale acestei activități sunt următoarele:

- au fost concepute mai multe scenarii noi pentru inteligență ambientă; accentul acestor scenarii a fost către funcționarea sistemului în ansamblul său, către adaptabilitatea și scalabilitatea sistemului, către rezolvarea problemelor, și către medii colaborative și publice, sau unde setările implică mai mulți utilizatori;
- a fost realizat un studiu asupra sistemelor multi-agent pentru dezvoltarea de aplicații de Inteligență Ambientă, observând caracteristicile care sunt oferite și tendința de a folosi fie puțini agenți complecși sau mulți agenți simpli, compensând reprezentări puternice și flexibilitate; am investigat, de asemenea, integrarea dependenței de context în aplicații de Inteligență Ambientă, observând echilibrul între complexitatea și puterea de reprezentare a informațiilor de context, pe de o parte, și descentralizarea sistemului, pe de altă parte; o mențiune specială este dedicată algoritmilor de potrivire de grafuri, cu accent pe potrivirea de grafuri etichetate;
- a fost realizat un studiu asupra auto-organizării în sistemele multi-agent, observând lipsa relativă de auto-organizare în sisteme formate din agenți cognitivi; un accent a fost pus, de asemenea, asupra mecanismelor prin care sunt obținuți emergenții doriți;

- am propus un model de diseminare a informațiilor într-un mediu distribuit pe baza unui set de măsuri de context care permit dezvoltatorului controlul răspândirii informațiilor într-un sistem multi-agent format dintr-un număr mare de agenți. Aceste măsuri de context sunt presiunea – controlul vitezei de răspândire; specialitatea – controlul direcției și a zonelor în care se răspândește informația; precum și persistența – controlul valabilității informațiilor;
- a fost dezvoltat un comportament al agentului care folosește mecanisme de auto-organizare (de exemplu, bucle de feedback pozitiv și negativ, un anumit nivel de aleator, etc.), care duce la o diseminare controlabilă a informațiilor la nivel global (nivelul sistemului), deși agenții individuali au doar cunoștințe locale și comunică doar cu vecinii lor imediați;
- am proiectat și implementat un mediu de simulare pentru sisteme multi-agent formate dintr-un număr mare de agenți, orientat spre simulări rapide, repetabile care rulează pe o singură mașină;
- am dezvoltat un format bazat pe XML pentru fișierele de scenariu, care permite testarea repetabilă de sisteme multi-agent mari, și care specifică structura agenților și momentul și natura evenimentelor, toți parametrii putând fi aleatori într-un interval definit și cu o abatere specificată;
- au fost proiectate și implementate instrumente de vizualizare pentru sistemele multi-agent mari, care permit vizualizarea evoluției sistemului în ansamblul său, precum și a evoluției agenților individuali; instrumentele includ, printre altele, vizualizarea valorii instantanee a unui parametru pentru fiecare agent al sistemului, vizualizarea evoluției în timp a unui parametru agregat peste întregul sistem, etc;
- am conceput un model pentru sisteme multi-agent dependente de context, în care topologia sistemului (relațiile de vecinătate) sunt legate de contextul agenților și de structura generală a contextului;
- topologia sistemului bazată pe context a fost validată prin proiectarea și dezvoltarea prototipului Ao Dai, în care agenții sunt asociați cu elemente de context (cum ar fi locuri, servicii și dispozitive) și relațiile ierarhice dintre agenți sunt mapate peste structura contextului; proiectul a fost implementat în CLAIM și a fost demonstrat la al 5-lea Workshop NII-LIP6, în iunie 2010 la Paris;
- am propus un formalism unificat pentru reprezentarea de informații de context, atât în interiorul cât și în afara agentului; formalismul este original și se bazează pe un graf, în care subgrafuri diferite reprezintă cunoștințele agenților, relațiile dintre agenții, și asocierea de agenți la containere (sau dispozitive);
- am conceput un formalism pentru șabloane de grafuri, care permite unui șablon să se potrivească peste o gamă mai largă de grafuri individuale, folosind noduri cu etichete nespecificate, precum și muchii etichetate cu expresii regulate;
- am dezvoltat un algoritm de potrivire a unui șablon de context peste un graf de context, care permite potriviri parțiale și produce subgraful care corespunde șablonului, parte șablonului care a fost potrivită, și o măsură numerică pentru partea de model care a fost potrivită;
- am îmbunătățit comportamentul agentului pentru diseminarea de informații

folosind interacțiuni locale pentru a integra topologia dependentă de context a sistemului și utilizarea de șabloane de context pentru recunoașterea situațiilor relevante;

- am simplificat semantica limbajului de programare orientată-agent CLAIM din punct de vedere al numărului și al categoriilor de primitive, și am simplificat sintaxa pentru a utiliza mai puține construcții și caractere speciale, având ca rezultat limbajul S-CLAIM care este mai simplu, mai curat, mai ușor de citit și mai ușor de interpretat;
- a fost integrată mobilitatea ierarhică a agenților JADE, într-un mod similar cu mobilitatea agenților CLAIM;
- am implementat clase și funcționalitate care să permită unui agent JADE să execute descrierea agentului extrasă din codul S-CLAIM, folosind, de asemenea, funcționalitatea pentru mobilitate ierarhică;
- am propus un formalism pentru reprezentarea liniară a grafurilor, și doi algoritmi pentru liniarizarea unui graf (pentru reprezentarea în text) și pentru extragerea de componente liniare dintr-un graf (pentru reprezentare grafică);
- a fost implementată o infrastructură de vizualizare care permite centralizarea logurilor agenților, precum și vizualizarea centralizată a topologiei dinamice a sistemului; vizualizarea folosește algoritmul pentru liniarizare de grafice;
- am dezvoltat un format bazat pe XML pentru specificarea de scenarii pentru sisteme multi-agent distribuite, asigurând o simulare facilă și repetabilă de sisteme multi-agent bazate pe JADE (și nu numai);
- am dezvoltat o platformă bazată pe sisteme multi-agent pentru aplicații de AmI, axată pe nivelul aplicație al inteligenței ambiante, folosind o reprezentare bazată pe grafuri bazate a informațiilor de context, și folosind limbajul de programare orientat agent S-CLAIM.

7.1 Perspective

Această lucrare este doar o fază, un început. Multe căi pentru dezvoltarea conceptelor pe care le-am introdus rămân deschise și doar prea puțin explorate. Implementările care au fost realizate merită să fie îmbunătățite și extinse, iar multe concepte au nevoie în continuare de testare în aplicații și scenarii întotdeauna mai aproape de viața reală. Unele dintre aceste potențiale ținte pentru viitor sunt prezentate în aceasta ultima secțiune a tezei.

Există mai multe perspective pentru platforma AmIciTy:Mi. Este o bună platformă pentru studiul sistemelor complexe formate dintr-un număr mare de agenți cu un comportament similar. Avantajele sale sunt viteza – pentru a permite realizarea de simulări multe – și instrumentele de vizualizare.

AmIciTy:Mi poate fi extins pentru a suporta agenți de diferite mărimi (capacitatea de a stoca mai multe sau mai puține informații). De asemenea, suportul de agenți care se deplasează în spațiu ar fi un plus. Fișierele de scenariu ar putea

fi îmbunătățite pentru a putea specifica dimensiunile și căile agenților. Studiul agenților în mișcare ar permite o mai bună perspectivă asupra modului în care se răspândește informația, într-un sistem mai realist.

Aceasta ar putea fi, de asemenea, folosită pentru a dezvolta noi măsuri de context, care ar fi la fel de simple dar, de asemenea, la fel de eficiente ca și cele deja dezvoltate. La fel, dezvoltarea de noi măsuri pentru evaluarea evoluției sistemului, și noi instrumente de vizualizare a sistemului.

Prototipul Ao Dai a condus deja la multiple urmări, atât în dezvoltarea teoretică, precum și în dezvoltarea platformei, și anume extinderea aspectelor considerate de context, pe de o parte, și dezvoltarea limbajului S-CLAIM și noua platformă, pe de altă parte.

Dar poate mai important, proiectul Ao Dai a fondat o colaborare între studenții din mai multe universități din Europa, Brazilia și Asia – echipa MAS de la Universitatea Pierre et Marie Curie (Paris 6), AI-MAS de la Universitatea "Politehnica" din București, institutul IFI din Hanoi și Universitatea PUC-Rio din Brazilia. Această colaborare este probabil să continue.

În ceea ce privește dependența de contextul (atât din interiorul cât și în afara agentului), credem că abordarea noastră are mult potențial, care nu a fost încă explorat în întregime. Ambele tipuri de integrare a contextului sunt noi contribuții, prin urmare numai prin teste intensive se va valida pe deplin adecvarea acestora la diverse aplicații.

Temporalitatea a fost doar puțin explorată. Experiența utilizatorului (sau a agentului), precum și activitățile planificate, ar putea fi formalizate ca elemente de context și utilizate ca atare.

Un alt aspect care are nevoie de o testare în aplicații practice și non-simulate este utilizarea proprietăților *characteristic* și *actionable* al șabloanelor, care au puterea de a crește foarte mult posibilitățile de aplicare a șabloanelor.

Foarte necesară în viitor este implementarea de scenarii mai multe, și mai variate, folosind topologia sistemului dependentă de context și șabloanele de context. Numai prin experiment se poate observa puterea reală a acestor reprezentări.

Platforma MAS pentru AmI trebuie să fie dezvoltată și extinsă în continuare; diferitele îmbunătățiri vor fi facile, deoarece se bazează pe componente bine implementate, care au un grad ridicat de generalitate. Aceasta se poate dovedi a fi nu numai un experiment AmI, dar de asemenea, o platformă foarte utilă și simplu de utilizat pentru implementarea și testarea de sisteme multi-agent. Limbajul de programare orientat-agent S-CLAIM este mai ușor de folosit decât CLAIM, și are o șansă bună de a deveni un limbaj pentru agenți utilizat pe scară mai largă.

Lista de publicații

Florea, A. M., Kalisz, E., and Olaru, A. (2009). Levels of emergent behaviour in agent societies. In Proceedings of CASYS'09, the 9th International Conference on Computing Anticipatory Systems, August 3-8, Liege, Belgium, pages 81-88. American Institute of Physics (AIP) Conference Proceedings. ISSN 0094-243X, ISBN 978-0-7354-0579-0 (ISI Proceedings).

Olaru, A., Marinica, C., and Guillet, F. (2009). Local mining of association rules with rule schemas. In Proceedings of CIDM 2009, the IEEE Symposium on Computational Intelligence and Data Mining, March 30 - April 2, Nashville, TN, USA, IEEE Symposium Series on Computational Intelligence, pages 118-124. (ISI Proceedings).

Marinica, C., Olaru, A., and Guillet, F. (2009). User-driven association rule mining using a local algorithm. In Cordeiro, J. and Filipe, J., editors, Proceedings of ICEIS 2009, the 11th International Conference on Enterprise Information Systems, May 6-10, Milan, Italy, pages 200-205. ISBN 978-989-8111-85-2 (ISI Proceedings).

Olaru, A. and Florea, A. M. (2009). Emergence in cognitive multi-agent systems. Proceedings of CSCS17, the 17th International Conference on Control Systems and Computer Science, MASTS Workshop, May 26-29, Bucuresti, Romania, 2:515-522. ISSN 2066-4451.

Olaru, A., Gratie, C., and Florea, A. M. (2009). Context-aware emergent behaviour in a MAS for information exchange. In Proceedings of ACSys09, 6th Workshop on Agents for Complex Systems, in conjunction with SYNASC 2009, September 26-29, Timisoara, Romania, pages 17-22.

Olaru, A., Gratie, C., and Florea, A. M. (2009). Emergent properties for data distribution in a cognitive MAS. In Papadopoulos, G. A. and Badica, C., editors, Proceedings of IDC 2009, 3rd International Symposium on Intelligent Distributed Computing, October 13-14, Ayia Napa, Cyprus, volume 237 of Studies in Computational Intelligence, pages 151-159. Springer. ISBN 978-3-642-03213-4 (ISI Proceedings).

Olaru, A., Gratie, C., and Florea, A. M. (2009). Measures of context-awareness for self-organizing systems. In Proceedings of EUMAS 2009, 7th European Workshop

on Multi-Agent Systems, Dec 17-18, Ayia Napa, Cyprus.

El Fallah Seghrouchni, A., Florea, A. M., and Olaru, A. (2010). Multi-agent systems: a paradigm to design ambient intelligent applications. In Essaaidi, M., Malgeri, M., and Badica, C., editors, Proceedings of IDC'2010, the 4th International Symposium on Intelligent Distributed Computing, volume 315 of Studies in Computational Intelligence, pages 3-9. Springer. (ISI Proceedings).

El Fallah Seghrouchni, A., Olaru, A., Nguyen, T. T. N., and Salomone, D. (2010). Ao Dai: Agent oriented design for ambient intelligence. In Proceedings of PRIMA 2010, the 13th International Conference on Principles and Practice of Multi-Agent Systems.

Olaru, A., El Fallah Seghrouchni, A., and Florea, A. M. (2010). Ambient intelligence: From scenario analysis towards a bottom-up design. In Essaaidi, M., Malgeri, M., and Badica, C., editors, Proceedings of IDC'2010, the 4th International Symposium on Intelligent Distributed Computing, volume 315 of Studies in Computational Intelligence, pages 165-170. Springer. (ISI Proceedings).

Olaru, A. and Florea, A. M. (2010). A graph based approach to context matching. Scalable Computing: Practice and Experience, 11(4):393-399. ISSN 1895-1767 (B+ Journal).

Olaru, A. and Florea, A. M. (2010). A graph based approach to context matching. In Proceedings of SYNASC 2010, 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania.

Olaru, A. and Gratie, C. (2010). Agent-based information sharing for ambient intelligence. In Essaaidi, M., Malgeri, M., and Badica, C., editors, Proceedings of IDC'2010, the 4th International Symposium on Intelligent Distributed Computing, MASTS 2010, the The 2nd International Workshop on Multi-Agent Systems Technology and Semantics, volume 315 of Studies in Computational Intelligence, pages 285-294. Springer. (ISI Proceedings).

Olaru, A., Gratie, C., and Florea, A. M. (2010). Context-aware emergent behaviour in a MAS for information exchange. Scalable Computing: Practice and Experience, 11(1):33-42. ISSN 1895-1767(B+ Journal).

Olaru, A., Gratie, C., and Florea, A. M. (2010). Designing Self-Organizing Cognitive Multi-Agent Systems. In Filip, F. G. and Enachescu, C., editors, Advanced Computational Technologies, Romanian Academy Press (in print)

Olaru, A., Gratie, C., and Florea, A. M. (2010). Emergent properties for data distribution in a cognitive MAS. Computer Science and Information Systems, 7(3):643-660. ISSN 1820-0214 (ISI Indexed Journal).

Olaru, A., El Fallah Seghrouchni, A., and Florea, A. M. (2011). Graphs and patterns for context-awareness. In Novais, P., Preuveneers, D., and Corchado, J. M.,

editors, Proceedings of International Symposium on Ambient Intelligence, University of Salamanca (Spain), 6-8th April, volume 92 of Advances in Intelligent and Soft Computing, pages 165-172. Springer. ISBN 978-3-642-19936-3, ISSN 1867-5662 (ISI Proceedings).

Olaru, A. and Florea, A. M. (2011). Context-aware agents for developing AmI applications. *Journal of Control Engineering and Applied Informatics*, 13(4). (in print) (ISI Indexed Journal).

Olaru, A. and Gratie, C. (2011). Agent-based, context-aware information sharing for ambient intelligence. *International Journal on Artificial Intelligence Tools*, 20(6):985-1000. (in print) (ISI Indexed Journal).

El Fallah Seghrouchni, A., Olaru, A., Nguyen, T. T. N., and Salomone, D. (2011). Ao Dai: Agent oriented design for ambient intelligence. In Proceedings of PRIMA 2010, the 13th International Conference on Principles and Practice of Multi-Agent Systems, number 7057 in *Lecture Notes in Artificial Intelligence*, pages 259-265. Springer (in print) (ISI Proceedings).

Olaru, A., Florea, A. M., and El Fallah Seghrouchni, A. (2011). An agent-oriented approach for ambient intelligence. *UPB Scientific Bulletin, Series C Electrical Engineering and Computer Science*. (awaiting review)

Bibliografie (selectie)

- [Augusto et al., 2010] Augusto, J., Nakashima, H., and Aghajan, H. (2010). Ambient intelligence and smart environments: A state of the art. *Handbook of Ambient Intelligence and Smart Environments*, pages 3–31.
- [Bolchini et al., 2007] Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F., and Tanca, L. (2007). A data-oriented survey of context models. *ACM SIGMOD Record*, 36(4):19–26.
- [Chen and Kotz, 2000] Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College.
- [Cook et al., 2009] Cook, D., Augusto, J., and Jakkula, V. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298.
- [Dey, 2001] Dey, A. (2001). Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7.
- [Ducatel et al., 2001] Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J. (2001). Scenarios for ambient intelligence in 2010. Technical report, Office for Official Publications of the European Communities.
- [El Fallah Seghrouchni, 2008] El Fallah Seghrouchni, A. (2008). Intelligence ambiante, les défis scientifiques. presentation, Colloque Intelligence Ambiante, Forum Atena.
- [Henricksen and Indulska, 2006] Henricksen, K. and Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2(1):37–64.
- [Heylighen, 2002] Heylighen, F. (2002). The science of self-organization and adaptivity. *The Encyclopedia of Life Support Systems*, pages 1–26.
- [Perttunen et al., 2009] Perttunen, M., Riekkki, J., and Lassila, O. (2009). Context representation and reasoning in pervasive computing: a review. *International Journal of Multimedia and Ubiquitous Engineering*, 4(4):1–28.

- [Ramos et al., 2008] Ramos, C., Augusto, J. C., and Shapiro, D. (2008). Ambient intelligence - the next step for artificial intelligence. *IEEE Intelligent Systems*, 23(2):15–18.
- [Serugendo et al., 2006] Serugendo, G. D. M., Gleizes, M.-P., and Karageorgos., A. (2006). Self-organization and emergence in MAS: An overview. *Informatica*, 30(1):45–54.
- [Suna and El Fallah Seghrouchni, 2004] Suna, A. and El Fallah Seghrouchni, A. (2004). Programming mobile intelligent agents: An operational semantics. *Web Intelligence and Agent Systems*, 5(1):47–67.
- [Weiser, 1995] Weiser, M. (1995). The computer for the 21st century. *Scientific American*, 272(3):78–89.