UNIVERSITATEA „BABEŞ - BOLYAI”
FACULTATEA DE ŞTIINŢE ECONOMICE ŞI GESTIUNEA AFACERILOR
CATEDRA DE INFORMATICĂ ECONOMICĂ

CONTRIBUŢII LA DEZVOLTAREA ARHITECTURILOR COLABORATIVE A DISPOZITIVELOR INTELIGENTE APLICATE ÎN ECONOMIE

PERVASIVE COMPUTING: COLLABORATIVE ARCHITECTURE APPLIED IN BUSINESS ENVIRONMENT

Rezumatul Tezei

Doctorand: Sebastian Presecan

Coordonator Ştiinţific: Prof. univ. Dr. Nicolae Tomai

Cluj-Napoca, November 2011

Abstract

În prezent, era computerelor personal e se apropie de sfârşit. Puterea computaţională este distribuită către diverse dispozitive de mici dimensiuni pe care utilizatorii le folosesc pentru a executa zilnic o multitudine de operaţiuni de rutină. Posesorii de dispozitive inteligente utilizează din ce în ce mai mult aplicaţiile mobile, în detrimentul celor de tip PC. Dispozitivele mobile sunt de asemenea capabile să ruleze aplicaţii care distribuie taskuri pentru a fi executate în cloud. Cercetarea de faţă propune implementarea unei sistem pervasive care să integreze diverse tipuri de dispozitive inteligente şi combine mobilitatea aplicaţiilor de pe telefoanele inteligente cu capacitatea de execuţie oferită de cloud. Soluția se dovedeşte a fi o soluţie viabilă pentru implementarea sistemelor pervasive și care utilizarea omniprezentă a puterii computaţionale.

Cuvinte cheie: pervasive computing, middleware, sisteme distribuite, aplicații mobile, cloud

1. Introducere

"Tehnologiile cele mai profunde sunt cele care dispar. Ele se întrepătrund în ţesătura vieţii de zi cu zi până când sunt imposibil de distins de aceasta"(Weiser M., 1991).. Cu această declaraţie, în 1991, Mark Weiser a descris viziunea lui despre computerul secolului 21. Sistemele de calcul au evoluat dincolo de PC-uri. Micro-computerele sunt în prezent încorporate în diverse dispozitive, precum telefoanele mobile, PDA-uri, diferite dispozitive de control, etc. Costul hardware-ului a scăzut în mod semnificativ, ceea ce a contribuit la răspândirea utilizării sistemelor de calcul.

Evoluția tehnologiilor de rețea a făcut posibilă ca sistemele de calcul să poată efectua activităţi pentru utilizator oriunde și oricând. În zilele noastre, calculatoarele au devinit tot mai omniprezente. Viziunea exprimată de Mark Weiser este în prezent mult mai aproape de realitate decât era de aşteptat vreodată.

1.1. Motivația

Sistemele mobile de calcul sunt astăzi o realitate, iar puterea de calcul este distribuită nu doar pentru telefoanele mobile. Telefoanele inteligente sunt omniprezente în zilele noastre. Proprietarii de telefoane inteligente folosesc aplicaţii mobile ca un înlocuitor pentru aplicațiile desktop. Ca urmare, există o nevoie tot mai mare de utilizare a sitemelor de calcul mobile în diferite domenii cum ar fi: medicina, afaceri, etc. Devine tot mai evident că există, de asemenea, o nevoie de a avea o arhitectura a unui sistem care să facă față acestor noi condiţii. Având în vedere experienţa mea ca și dezvoltator de software şi arhitect şi faptul că am fost implicat în proiectarea şi implementarea sistemelor distribuite de mari dimensiuni pentru mai mult de 12 de ani, aceste noi progrese tehnologice m-au provocat să încerc şi să defineac o nouă arhitectură pentru noua generaţie de sisteme mobile omniprezente.

1.2. Problematica de cercetare

Având în vedere stadiul actual al progresului tehnologic când puterea de calcul este răspândită în diferite tipuri de dispozitive, apare ca și o nouă provocare nevoia de a

crea un mediu în care aceste dispositive inteligente să colaboreze pentru realizarea sarcinilor primite de la utilizatori. Pentru moment, metoda veche de a face aceste dispozitive să colaboreze presupune conectarea lor la diferiți furnizori de servicii, în scopul de a obţine informaţii și de a le afișa utilizatorului.

Dispozitivele inteligente din zilele noastre sunt destul de avansate pentru a indentifica colaboratori care să îi ajute la îndeplinirii sarcinilor primite de la utilizatorilor sau de a delega aceste sarcini unor computere mult mai puternice. Având în minte toate aceste aspecte, mi-am pus următoare întrebari:

- Este posibil să îmbunătăţim aplicaţii smartphone-ul să detecteze și să se adapteze la mediul în care rulează?
- Care sunt modalităţile posibile pentru a face dispozitivele inteligente să comunice direct între ele?
- Care este cea mai bună abordare pentru a permite dispozitivelor inteligente de a transfera taskurile primite de la utilizatori la computere mult mai puternice?
- Cum ar trebui să fie proiectat un astfel sistem care să asigure securitatea și confidenţialitatea într-un mediu atât de dinamic ca și cel mobil?
- Cum este posibil să integrezi toate tipurile diferite de dispozitive inteligente care rulează pe sisteme de operare diferite într-un sistem unic?

Pe parcursul cercetărilor actuale am căutat răspunsuri la toate aceste întrebări. Prin punerea împreună a tuturor acestor răspunsuri am reuşit să proiectez un middleware care ar putea fi utilizat pentru punerea în aplicare a oricarui sistem pervasive. Scopul cercetării actuale este de a defini o arhitectura scalabilă pentru a fi utilizată în punerea în aplicare a diferitelor cerinţelor funcționale, cu efort cât mai mic pentru dezvoltatorii de aplicaţii. Middleware propus ar trebui să ajute ca folosirea aplicațiilor mobile să devină omniprezentă.

1.3. Obiective

Unul dintre obiectivele lucrării de faţă este de a înţelege modul în care soluțiile existente susţin sistemele de calcul pervasive şi să le analizeze din perspectiva implementării unui portal educaţional omniprezent.

Rezultatul cercetării curente este o arhitectură scalabilă pentru un sistem mobile omniprezent. Vom considera ca și domeniu aplicativ un portal educaţional dintr-un campus universitar. Utilizarea unui astfel sistem este menită să fie omniprezentă, astfel încât utilizatorii să nu fie sunt deranjați de către sistem, cu excepţia situaţiei când, după consumarea tuturor posibilităţile existente, aceasta nu poate stabili de la sine modalitate de continuare execuției. Chiar dacă arhitectura propusă a fost proiectată pentru a îmbunătăţi un portal educaţional omniprezent, noi considerăm că este suficient de generică pentru a putea fi folosită la implementarea oricărui tip de sistem pervasive.

1.4. Contribuții

Teza de doctorat ridică mai multe probleme de noutate în domeniul Pervasive Computing. Principalul domeniu de studiu şi de cercetare cuprinde conştientizare de context, invizibilitatea, securitate şi mobilitatea aplicațiilor. Toate aceste aspecte au fost analizate din perspectiva de a construi un sistem pervasive. Contribuțiile majore sunt următoarele:

- identificarea problemelor majore în procesul de implementare a unui sistem pervasive şi definirea criteriilor de evaluare, care ar putea fi utilizate pentru o analiza a middleware-urilor folosite pentru implementarea sistemelor pervasive actuale
- proiectarea unei soluţii inovatoare care ajuta sistemul să fie sensibil la modificările de context şi care permite colectarea informațiilor despre context de la mai mulți furnizorii. Acestea pot fi utilizate de către aplicaţiile mobile, să monitorizeze mediul înconjurător şi să adapteze caracteristicile aplicaţiei şi funcţionalităţi la schimbările context, într-un mod invizibil;
- conceperea unei soluţii de ultimă oră pentru a atinge mobilitatea datelor și a aplicațiilor prin distribuirea de servicii pentru aplicaţii în Cloud într-un mod transparent pentru dezvoltatorul de aplicații.
- folosirea un canal de comunicare bazat pe standarde de-facto, pentru a integra diferite dispozitive în sistem.
- proiectarea arhitecturii unei soluții pentru implementarea oricărui tip de sistem omniprezent care se confruntă toate provocările identificate pentru un astfel de sistem.

2. Domeniul de cercetare

Acest capitol este axat în principal pe prezentarea conceptelor cheie ale Pervasive Computing. Nu s-a dorit realizarea unei monografii detaliate a Pervasive Computing, ci mai degrabă să se prezinte conceptele-cheie, care sunt necesare pentru a efectua analiza a unei arhitecturi a unui sistem omniprezent. Sunt prezentate provocările Pervasive Computing, unele dintre sistemele care sunt capabile să rezolve aceste provocări, precum şi direcţiile principale de studiu din cadrul acestui domeniu de cercetare.

2.1. Evoluția Pervasive computing

Declaraţia vizionară a lui Mark Weiser(menţionatată mai sus) descrie ceea ce se aşteaptă de la pervasive computing: utilizatorilor au acces la sistemul de calcul oricând și oriunde.

Începând cu definiţia: "Un sistem distribuit este un sistem de procesare a informaţiilor, care conţine un număr de calculatoare independente care cooperează unul cu altul printr-o reţea de comunicaţii, în scopul de a atinge un obiectiv specific" (Bapat, 1994) precizează că "pervasive computing", defineşte un nou context pentru interacţiunea dintre utilizatorii finali şi sisteme distribuite.
Definiția de către A.S. Tanenbaun a sistemelor distribuite: *"O colecţie de calculatoare independente care se afişează utilizatorilor săi ca un sistem unic şi coerent"* (Tanenbaun & Steen, 2006) a prezis omniprezenţa sistemelor de calcul zilele noastre.

Având în vedere evoluţia computerelor şi interacţiunea dintre utilizatori şi sistemul de calcul, putem identifica trei etape principale:

- mainframe - care poate fi definită ca și *un sistem cu utilizatori multipli;*
- calculator personal - care poate fi definită ca și *un utilizator, un singur sistem;*
- sistem de calcul mobil și omniprezent - care poate fi definită ca și *un utilizator, mai multe sisteme.*

Pervasive computing a schimbat modul în care oamenii interacţionează cu computerele. Ideea-cheie din spatele pervasive computing este de a folosi în mod aproapre inconştient diferite dispozitive pentru realizarea sarcinilor noastre de zi cu zi. Aceste dispozitive se coordonează între ele cu scopul de a oferi utilizatorilor un acces universal la informaţii şi pentru a le acorda sprijin în îndeplinirea sarcinilor zilnice.

2.2. Provocările sistemelor pervasive

Având în vedere viziunea lui M. Weiser este clar că un sistem pervasive trebuie să fie distribuit şi trebuie să sprijine mobilitatea utilizatorului şi a aplicațiilor.

În

Figure 2.1, M. Satyanarayanan descrie principalele caracteristicile ale unui sistem pervasive.

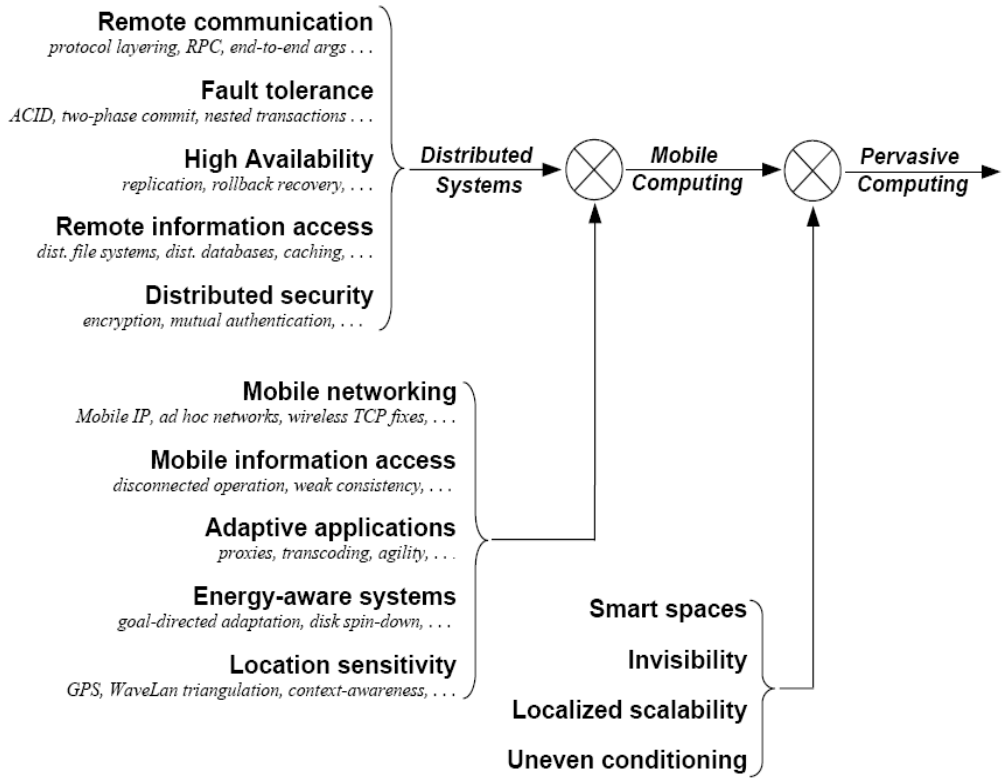

Figure 2.1 - Taxonomy of Computer Systems Research Problems in Pervasive Computing (Satyanarayanan, Pervasive Computing: Vision and Challenges, 2001)

2.3. Sisteme pervasive

Chiar dacă pervasive computing este un domeniu relativ nou, există deja diferite propuneri pentru arhitecturi ale unor sisteme pervasive. În cercetarea curentă, am selectat pe acelea care sunt reprezentative pentru acest domeniu de cercetare. Selecţia a fost făcută ţinând cont de aprecierea la nivel mondial a soluției şi de diferitele taxonomii folosite pentru a defini aceste arhitecturi.

Scopul evaluării este de a analiza modul în care aceste sisteme îndeplinesc cerinţele definite mai sus, de a identifica punctele forte şi punctele slabe ale sistemelor existente şi, de asemenea, de a identifica posibile direcții de cercetare.
Sistemele analizate sunt:

- GAIA-dezvoltat de Universitatea din Illinois;
- AURA-dezvoltat de Carnegie Mellon University;
- PICO - dezvoltat de Universitatea din Texas;
- One.world - dezvoltat de Universitatea Washington;
- SODA

2.3.1. Gaia

Gaia este un meta-sistem de operare construit ca și o infrastructura middleware distribuită care coordonează entităţile software şi dispozitivele conectate printr-o reţea eterogenă și care sunt situate într-un spaţiu fizic (Roman, Hess, Cerqueira, & Campbell, 2002). Gaia este conceput pentru a sprijini dezvoltarea şi executarea de aplicaţii portabile pentru *active spaces* – mini sisteme pervasive, în care utilizatorii interacţionează cu mai multe dispozitive şi servicii simultan. Gaia expune servicii pentru a interoga, a accesa, şi de a folosi resursele existente şi oferă un cadru pentru dezvoltarea aplicații mobile sensibile la modificările de context.

2.3.2. Aura

Ca și o consecinţă a legii lui Moore, în zilele noastre este clar că hardware-ul nu mai este gâtuire ci modul în care utilizatorul interacţionează şi utilizează sistemele de calcul.

Sistemul Aura acţionează ca un proxy pentru *utilizatorul mobil* pe care îl reprezintă: atunci când un utilizator intră într-un mediu nou, modul Aura assignat lui caută resursele adecvate pentru a sprijini *sarcina utilizatorului.* În plus, un modul Aura observă şi reacţionează la modificările de mediu. (Garlan, Siewiorek, şi Smailagic, 2002), (Aura, 2002)

2.3.3. One.World

Arhitectura One.World a fost creată cu scopul de a oferi o soluție fiabilă uneia dintre cele mai importante provocări ale sistemelor pervasive, care este "accesul la informaţii oricând şi oriunde". *Arhitectura One.World oferă un cadru integrat, cuprinzător pentru construirea de aplicaţii pervasive. Ea vizează aplicaţii care se adaptează automat la medii de calcul extrem de dinamic, şi include servicii care uşurează pentru dezvoltatorii de aplicații, gestiunea schimbărilor de mediu.* (Grimm R., 2004)

2.3.4. PICO

Obiectivul PICO este de a răspunde cerinţelor aplicaţiilor din domenii cum ar fi telemedicina, militare, şi de gestionare a crizelor care cer automatizate, servicii continue şi colaborare proactivă în timp real între dispozitive şi agenţi software în medii dinamice, eterogene. (Kumar, Shirazi, şi Singhal, 2003)

Arhitecții PICO şi-au imaginat un sistem alcătuit din softuri inteligente autonome numite *delegents* şi dispozitive hardware numit *camilenus. Camilenus* pot fi de diferite tipuri şi complexități. Acestea sunt toate interconectate folosind protocoale şi tehnologii diferite. Un *delegent* este un software inteligent care funcţionează în numele unui *camilenus* sau în numele utilizatorului. De exemplu, un delegent poate aduna informaţii la nivel local sau de la distanţă, cu scopul de a colabora cu alte delegents şi forma o comunitate de calcul.

În general, sistemul PICO oferă o soluție viabilă bazată pe paradigma agenților mobili pentru implementarea unui sistem pervasive.

2.3.5. SODA

Pentru a construi un sistem generalizat după cum am văzut mai sus, există multe provocări care trebuie să fie depăşite. Pentru a le satisface pe toate, sistemul riscă să devină extreme de complex şi costisitor. În scopul de a influenţa costul de integrare, dezvoltare şi pentru a obţine o bună scalabilitate, o soluţie ar fi să se adopte SOA în pervasive computing.

Service Oriented Architecture (SOA) este un set de principii care definesc o arhitectura în care este componentele sunt slab cuplate şi este formată din furnizorii de servicii şi consumatori de servicii care interacţionează în conformitate cu un contract sau interfaţă. (Mansukhani, 2005)

SODA este o adaptare a arhitecturii orientate spre servicii (SOA), care integrează sisteme de afaceri printr-un set de servicii care pot fi refolosite şi combinate pentru a aborda schimbarea priorităţilor de afaceri. Serviciile sunt componente software cu interfeţe bine definite, iar acestea sunt independente de limbajul de programare si platforme de calcul pe care rulează. (Koch, 2005) Astfel, scopul de SODA este de a integra sistemul de dispozitive în sistemele software pentru mediul enterprise. Prin acest demers, dezvoltatorii vor putea folosi dispozitivele încorporate şi senzori ca şi orice alt serviciu. (Sumi, 2006)

2.4. Observații finale

Pervasive Computing este o evoluție naturală a sistemelor distribuite. Utilizatorii sistemelor pervasive sunt capabili de a folosi diferite aplicaţii care rulează pe o gamă largă de dispozitive şi sisteme. Distribuţia execuției sarcinilor şi caracterul eterogen al mediului de funcţionare face proiectarea unui sistem pervasive o sarcină destul de complexă.

Am identificat principalele provocări arhitecturale care trebuie să fie depăşită de orice sistem îndeplinirii sarcinilor generalizată: context-conştientizare, invizibilitatea, serviciu de căutare, integrare, acces la informaţii mobile, scalabilitate, securitate şi dezvoltare şi implementare. Următoarele sisteme: *Gaia, Aura, One.World, Pico şi soda* au fost analizate din perspectiva implementării provocărilor mai sus identificate

arhitectural. Sistemele prezentate furnizează arhitectura puternică, abordarea unor provocări.

Principalele rezultate de la acest capitol au fost publicate pe (Presecan S., 2007), (S Presecan, 2008)

3. Context-awareness

Capitolul curent defineşte conceptul de "context-awareness". De asemenea, identifică şi descrie în detaliu principalele provocări presupuse de procesul de implementare a sistemelor care sunt conştiente de context. La sfârşitul capitolului, este prezentată propunerea mea pentru o mini architectură a unui sistem conştient de context care își propune să implementeze provocările indentificate.

3.1. Conceptul de Context-awareness

Un sistem conştient de context se adaptează în funcţie de locul de utilizare, de persoane și obiectele aflate în apropiere, de dispozitivele accesibile, precum şi modificări acestora de-a lungul timpului. Un sistem cu aceste capacităţi monitorizează mediul în care se execută şi reacţionează la schimbările de mediu.(Schilit, Adams, şi Dorim, 1994)

Dispozitivele conștiente de context și de aplicaţiile trebuie să răspundă la schimbările din mediul într-un mod inteligent, în scopul de a îmbunătăţi comportamentul sistemului de calcul raportat la utilizatori pe care îi deservește. Aplicațiile conștiente de context tind să fie aplicaţii mobile pentru motive evidente:

- contextul utilizatorului fluctueaza cel mai frecvent atunci când un utilizator este mobil.
- necesitatea de comportament conştient de context este mai mare în mediul mobil (Adelstein, Gupta, Richard, si Schwibert, 2005).

3.2. Provocările Context-awareness

În teorie, un dispozitiv inteligent poate să obţină informaţii de context folosind o mare varietate de senzori, camere video şi microfoane, care pot fi încorporate în aparat sau în mediul înconjurător. În practică, aceasta este o sarcină foarte dificilă, dacă luăm în considerare nevoia de integrare a diferite tipuri de mici dispozitive şi tehnologii, mobilitatea utilizatorului şi puterea de calcul resusă a dispozitivelor inteligente.

Următoarele provocări pot fi considerate importante pentru middleware-ul unui sistem conştient de context:

- descoperirea sursei de furnizorii de context.
- interogarea pentru a obţine informaţii contextuale.
- prelucrarea informaţiilor despre context fără a afecta performanțele normale a sistemului.
- adaptabilitatea sistemului la schimbare a valorilor context.
- integrarea diferitelor tipuri de furnizori de context şi de canale de comunicare.

3.3. Arhictectură context-aware

Informaţiile despre context pot fi obținute cel puţin din următoarele tipuri de surse:

- senzori conectați direct la un dispozitiv inteligent
- senzori conectați la un calculator.
- generatori de context care generează informaţii context.

Arhitectura următoare este propusă în scopul de a sprijini conştientizarea contextul de către un sistem pervasive. Figura 3.1 descrie principalele componente necesare pentru a construi un sistem distribuit care să fie conștient de context.

O componentă cheie este *ContextService*. Acesta este responsabil pentru a obţine toate informaţiile despre mediul în care rulează o aplicaţie instalată într-un dispozitiv inteligent. Acesta preia informaţii contextuale de la senzori direct conectați la acesta sau le primeşte de la "buddy" ContextService.

"Buddy" ContextServices constau dintr-o colecţie de servicii distribuite pe maşini diferite, care colaborează în scopul de a prelua toate informaţiile necesare despre mediul în care rulează aplicațiile mobile. Aceste servicii poate fi definite static sau dinamic. Pentru configurarea statică, putem defini o listă a parametrilor QoS care trebuie să fie utilizată atunci când serviciile sunt preluate din DiscoveryService. O configurare dinamică presupune că serviciile înceapă să lucreze împreună o dată ce nu pot obține direct informațiile cerute de utilizator sau nu mai pot răspunde în timp util utilizatorilor.

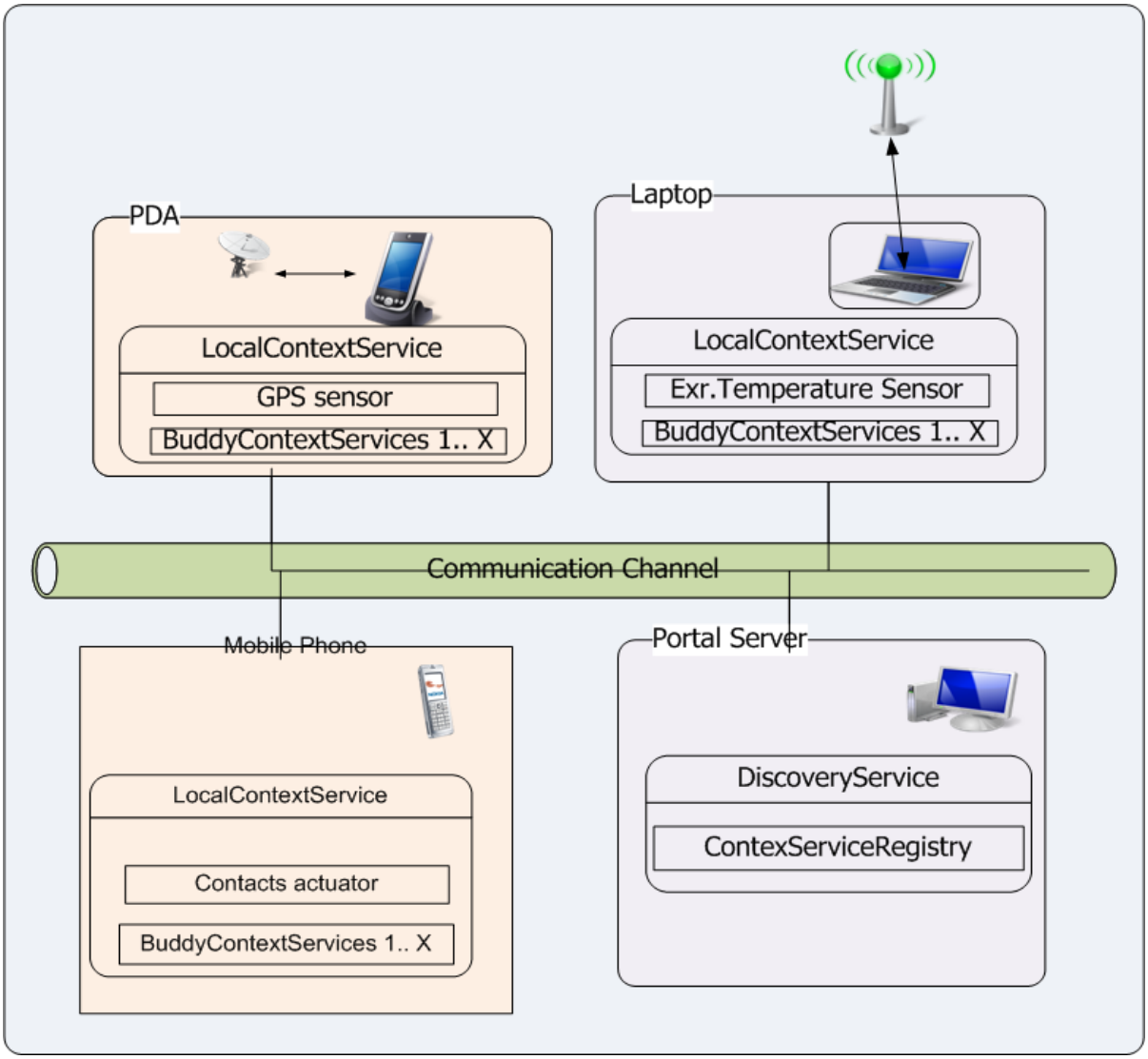

Figure 3.1 - Architecture overview

BuddyContextService este un proxy inteligent, care este creat de la *DiscoveryService*. *DiscoveryService* analizează parametrilor QoS transmise de către BuddyContextService în timpul cererilor pentru un nou ContextService şi dă clientului acel ContextService care este cel mai potrivit pentru capabilităţile pe care le solicită. Proxy-ul este considerat a fi inteligent, deoarece DiscoveryService ar putea să-l înlocuiască în cazul în care nu oferă funcţionalitatea aşteptată. În acest fel, atunci când QoS nu mai sunt asigurate de Buddy-ul actual, DiscoveryService este responsabil să-l înlocuiască cu un nou Buddy. În consecinţă, sistemul asigură scalabilitatea locală, care este o cerinţă importantă pentru sistemele omniprezente.

Fiecare ContextService poate fi conectat direct la unul sau mai mulți senzori şi generatori de context. The ContextService de la dispozitivul legat la sensor sau care înglobează senzorul va comunica direct cu senzorul, și va publica valorile senzorului pentru întregul sistem.

Generatoarele de context sunt folosite pentru a crea informaţii contextuale în sistem la cererea utilizatorilor. Generatoare nu se simt cu adevărat de mediu, dar ele oferă informaţii care pot ajuta la intenţia utilizatorului. De exemplu: generatoarele prezenţă, furnizează informații altor utilizatori atunci când utilizatorul curent se alătură sau părăsește o comunitate.

3.4. Concluding remarks

Capitolul curent are drept scop prezentarea caracteristicii context-awarenes, una dintre cele mai importante provocări pentru un sistem pervasive: *integrarea furnizorilor de context diferit, extensibilitate, de distribuţie şi de cooperare, bazată pe evenimente de infrastructură, domeniu interfaţă simplă mobile, interogare simplu, descoperirea dinamică şi de înregistrare.*

A fost proposusă o architectură de middleware care să resolve provocările legate ce context-awareness într-un sistem pervasive. Middleware propus este bazat pe ideea de a considera o distribuire a furnizorilor de context şi posibilitatea de a-i integra într-un sistem care este capabil să le monitorizeze parametrii QoS. În cazul în care middleware-ul observă că unei furnizori de context nu funcţionează în mod normal, îi înlocuieşte cu furnizorii, fără a întrerupe activităţile utilizatorului. Informațiile despre context pot fi preluate de la serviciile contextul învecinate, care pot fi localizate în reţele aflate în apropiere, şi nu numai de la senzorul conectat direct, astfel cum este cazul middleware-urilor existente. Middleware permite furnizarea informațiilor despre context în mod trasparent, astfel încât clientul poate solicita parametri contextuali fără să ştie de unde acest context. Mecanismele de localizare ale furnizorile și de interogare a servicilor furnizoare de context sunt asigurate de către middleware și sunt ascunse utilizatorilor.

Rezultatele din acest capitol sunt prezentate pe scurt în următoarele studii: (Presecan & Tomai, 2009), (Presecan S., 2009), (Presecan S., 2009a)

4. Invizibilitate

Context-awareness este primul pas în atingerea "omniprezenţei", atunci când se utilizează un sistem de calcul. Un sistem de conştient de context observă mediul înconjurător şi se poate adapta la modificările acestuia. Modurile în care aceasta reacţionează ar putea face ca sistemul să fie într-adevăr pervasive. Capitolul curent definește conceputul de invizibilitate în contextual sistemelor pervasive. Modalitățile de realizare a "invizibilitatii" sunt prezentate detaliat şi este propusă o soluţie de realizare a invizibilității, cu scopul de a îmbunătăţi soluţiile disponibile în prezent.

4.1. Conceptul de invizibilitate

Având în vedere obiectivul de a avea o reacţie proactivă la schimbarea de context, sistemele pervasive tind să fie enervante prin notificarea utilizatorului la orice schimbare context. Adaptarea la schimbările de mediu ar trebui să se facă automat, fără a perturba prea mult utilizatorul. Acesta este rolul middleware-ului de a facilita adaptarea, care poate implica adaptarea componentelor individuale, a software-ul şi / sau reconfigurare a componentelor prin adăugarea, eliminarea sau înlocuirea acestora. În ceea ce privește sistemele pervasive, adaptarea la modificarile de context tinde să fie mai degrabă application-aware şi mai puţin user-aware.

Având un astfel de sistem, interacţiunea între aplicaţia client şi sistem devine invizibilă pentru utilizator; cele mai multe dintre măsurile de adaptare sunt efectuate de către aplicaţia client, fără a întrerupe operaţiunile efectuate de către utilizator. Provocarea constă în a găsi metoda adecvată de a integra aceste tehnologii eterogene într-un singur sistem, în scopul de a permite conectivitate diferitelor tipuri de dispozitive.

4.2. Provocări

Adaptabilitatea sistemul pervasive ar trebui să fie configurabilă, astfel încât numai anumite condiţiile specifice ar putea duce la interacţiuni cu utilizatorul. Utilizatorul ar trebui să poată configura sistemul în aşa fel încît să reacţioneze în diferite moduri la schimbarea contextului.

În scopul de a asigura invizibilitate şi adaptabilitatea aplicațiilor, middleware trebuie să facă faţă următoarelor provocări:

- *Să ofere suport pentru configurare statică* pentru regulile de reacţie - utilizatorul ar trebui să poată să configureze, folosind un limbaj de configurare simplu, reacţiile la schimbarea contextului.
- *Să ofere suport* pentru *configurare dinamică* a regulilor de reacţie - aplicaţiile externe ar trebui să fie în măsură să înregistreze şi să configureze normele de reacţie bazată pe raţionament logic simplu. Configurarea ar trebui să fie făcută folosind un limbaj descriptiv simplu, care este deschisă pentru a adăuga noile condiţii de filtrare.
- *Prelucrare regulilor de reacţie nu ar trebui să afecteze performanţa sistemului în ansamblu.* Astfel, sistemul ar trebui să proceseze evenimentele şi de a executa regulile de reacţie într-un mod inteligent, pentru a evita reducerea performanţei sistemului.

4.3. Detalii de proiectare

Pornind de la aceste provocări, un motor decisional simplu, a fost conceput cu scopul de a sprijini adaptabilitatea sistemului. Figura 4.1 descrie principalele clase necesare pentru a pune în aplicare un motor decizional.

Clasa cheie este *Rule.* Este utilizată pentru a modela o regulă de reacţie. Acesta conţine un *EventFilter* şi de *Action* care trebuie să fie realizată odată ce evenimentul a trecut de filtrul. Rule, EventFilter şi Action urmează modelul ECA (Even-Condition-Action) (Ipina & Katsiri, 2001)

Rule se modelează printr-o simplă directivă IF-THEN-ACTION:

If EventFilter Then Action

Bazat pe această directivă, este destul de simplu pentru a construi un set de configuraţii statice sau dinamice pentru reacţiile de sistem. Utilizatorul poate defini cu uşurinţă preferinţele proprii pentru reacţii la schimbările în context.

Odată ce regula este definită, acesta trebuie să fie înregistrată în RuleManager. Managerul suportă 2 moduri de înregistrare a regulilor:

- înregistrare dinamică - în cazul în care aplicaţia creează regula şi apoi o adaugă la manager.
- Înregistrare statică - la start-up, atunci când RuleManager citeşte o configuraţie statică, caz în care regulile sunt stocate, folosind un limbaj descriptiv simplu.

Un limbaj descriptiv simplu ar putea arata astfel:

regulă: = nume condiţiile de acţiune

*condiţii: = Stare **

Stare: = contextParam operator valoare

operatorul: = = |> | <|> = | <=

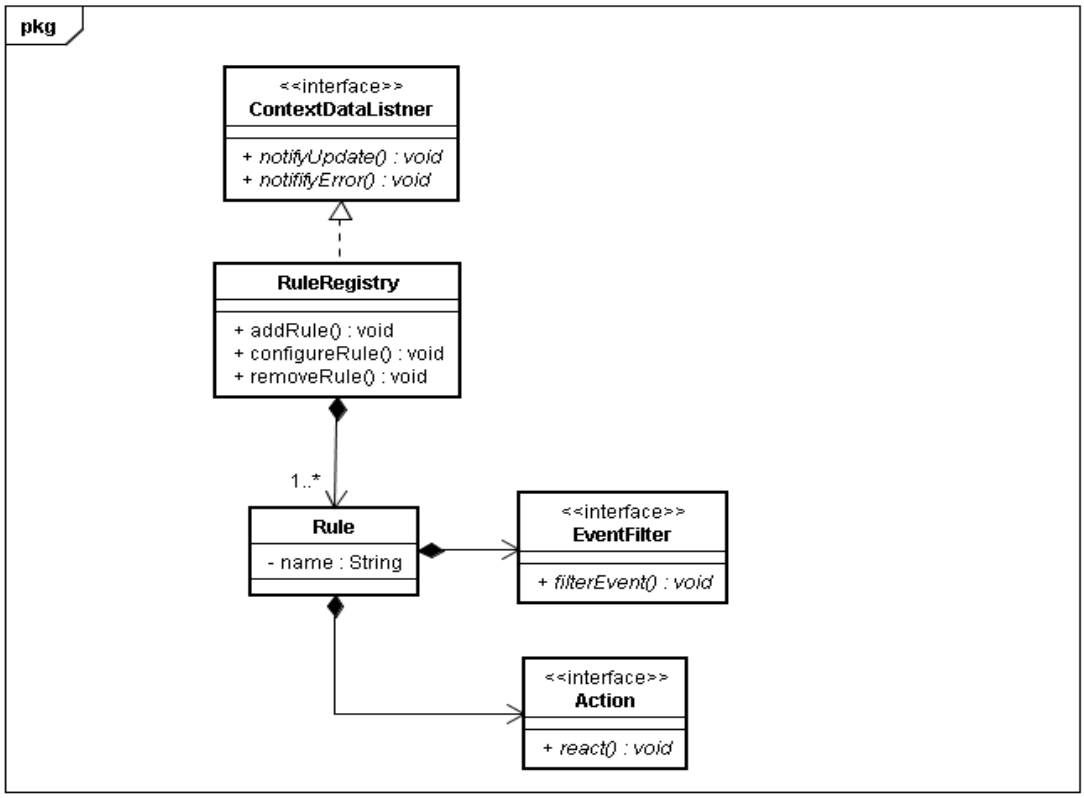

Figure 4.1 - Rule Engine Class Diagram

RuleManager este un ContextDataListener; care în timpul start-up găseşte ContextService corespunzătoare şi se înregistrează pe sine ca şi ContexDataListener, pentru a putea primi orice o notificare cu privire la schimbarea contextului. Când un eveniment este primit de la ContextService, RuleManager caută prin toate regulile înregistrate, cu scopul de a verifica pentru care dintre ele evenimentul trece EventFilter configurat. În cazul în care găseşte unele reguli pentru care evenimentul este selectat de către EventFilter, aceasta începe să execute acţiunile definite de aceste reguli. Pentru a evita scăderea puterii de calcul, aceste acţiuni pot fi executate sincron sau asincron, bazat pe o configuraţie definită de utilizator.

4.4. Observații finale

The outcomes of this chapter are summarized in the following studies: (Presecan S. , 2009), (Presecan & Tomai, 2009)

Orice sistem pervasive trebui observe şi să se adapteze la schimbările din mediul. Capitolul curent a descris un mod simplu de a reactiona la schimbarile de mediu. Middleware s-ar putea adapta la schimbările în funcţie de preferinţele utilizatorului sau pe baza strategiei de reacţie predefinite. Design-ul middleware propus se îmbină cu arhitectura definită pentru implemenarea unui sistem conştient de context. Prin îmbinarea caracteristicilor de context-aware şi invizibilitate, middleware propus ajută dezvoltatorii pentru a implementa aplicații care sunt suficient de inteligente pentru a se adapta la mediu eterogen şi dinamic specifice unui sistem pervasiv.

Rezultatele din acest capitol sunt prezentate pe scurt în următoarele studii: (Presecan S., 2009), (Presecan & Tomai, 2009)

5. Mobilitatea aplicațiilor

Într-un mediu pervasive, utilizatorii şi dispozitivele sunt într-o mişcare continuă. Informaţiile sunt colectate din diferite surse, a căror amplasare s-ar putea schimba şi chiar mai mult, conexiunea de date este în mişcare de la un furnizor la altul. În acest context, mobilitatea aplicațiilor şi a datelor este una din caracteristicile cheie care trebuie să fie susţinute de un middleware pervasive.

Un concept nou, denumit "cloaudable application" este introdus cu scopul de a defini un nou mod de realizare a mobilității aplicațiilor şi a datelor. Secţiunea finală a prezentului capitol prezintă o soluţie concretă pentru implementarea cloudable applications.

5.1. Mobile cloudable applications challenges

Sistemele pervasive au ca şi scop să ajute aplicaţiile să ruleze în medii mobile, adaptându-le la schimbările mediului. Orice aplicație pervasive ar trebui să fie capabilă să ruleze pe dispozitive mobile, care au o putere de calcul limitată şi care nu au conexiune la Internet foarte bună. Datorită resurselor limitate şi datorită scopului de a fi un înlocuitor real pentru calculatoare personale, dispozitivele mobile ar trebui să poată rula aplicaţii care să permit distribuirea sarcinilor care în cloud pentru a fi executate.

Pentru a face o distincţie clară între aplicaţiile mobile normale (care rulează pe dispozitive mobile şi folosesc programatic resursele disponibile pe acele dispozitive sau consumă conținut de pe Internet) şi noile aplicaţii mobile, care se ocupă de cererile utilizatorului prin divizarea şi executarea unor taskuri care pot fi distribuite pentru execuție, sugerăm să numim ultimul tip de aplicaţii mobile: *"mobile cloudable applications"*.

Mobile cloudable application - este o aplicaţie mobilă care este capabilă de a migra în cloud, cu scopul de a îndeplini sarcinile utilizator într-un mod mai bun, prin utilizarea puterii de calcul disponibile în cloud.

Orice framework utilizat pentru dezvoltarea de aplicaţii mobile cloudable care sunt utilizate în mediu pervasive ar trebui să îndeplinească următoarele cerinţe:

- *susține dezvoltarea de aplicații conștiente de context* - utilizatori mobili sunt caracterizați prin contexte diferite: local, comportamental, social, etc. Aplicaţiile pervasive folosesc aceste contexte pentru a înţelege obiceiurile utilizatorului şi să își adapteze execuția la preferinţelor utilizatorului.
- *reacţionează la schimbările din mediu* - în timpul executării lor, aplicaţiile trebuie să reacţioneze cu privire la modificările mediului. Aplicaţiile mobile cloudable îmbrățișând strategia "reacţionează-şi-adaptează", se adaptezea la schimbările mediului.
- *distribuţie transparentă a execuției în cloud* - dezvoltatorii de aplicaţii nu ar trebui să fie constrânși să utilizeze bibliotecile particularizate pentru a folosi serviciul de distribuţie. Distribuirea sarcinilor în cloud ar trebui să fie transparentă pentru dezvoltator de aplicaţii şi ar trebui să fie realizată prin utilizarea framework-ului.
- *Mobilitate pentru aplicații şi date* - codul scris pentru o aplicație mobile cloudable ar trebui să fie migrat transparent în cloud. Codul şi date ar trebui să migreze dynamic la rulare; migrarea realizându-se cu scopul de a utiliza resursele dispozitivului mobil cât mai optim.
- *Elasticitatea aplicaților* - serviciile de cloud vor fi folosite la cerere doar în cazul în care resursa locală nu ar fi în măsură să îndeplinească nivelul aşteptat QoS. Framework-ul ar trebui să măsoare QoS şi ar trebui să reacţioneze la lipsa de resurse, prin distribuirea de execuție în cloud.

Aplicaţiile mobile cloudable sunt menite să utilizeze resursele dispozitivului mobil optim şi atunci când dispozitivul este lipsit de putere de calcul sau în cazul în care executarea sarcinilor în cloud este mai optimă, ele își distribuie executarea sarcinilor în mod transparent în cloud.

5.2. Framework architecture

Cu scopul de a îndeplini provocările aplicaților mobile cloudable, arhitectura framework-ului descris în Figura 5.1 ar putea fi folosite pentru dezvoltarea unei noi generaţii de sisteme pervasive.

Principalele componente ale arhitecturii propuse sunt:

- *Aplicaţia mobilă* – Aplicația este dezvoltată folosind framework-ul şi limbajul de programare specific unei platformei mobile: Java pentru Android, Objective C pentru iPhone / iPad, C# pentru WP7. Aplicația mobilă interacţionează cu utilizatorul folosind o interfaţă de utilizator, care este specifică platformei și dispozitivului mobil. Pentru a îndeplini cererile utilizatorilor, aplicația mobilă utilizează diferite servicii situate pe dispozitive mobile. Serviciile sunt fie servicii de sistem sau servicii personalizate.
- *LocalServices* – sunt implimentarea unor sarcini granulare utilizate pentru a îndeplini cererile utilizatorilor. Servicii ar putea rula pe dispozitivul mobil local, dar ar putea fi, de asemenea, rulate în cloud. Aceste servicii ar trebui să implementeze bussiness logic și nu ar trebui să depindă de interfaţa cu utilizatorul.
- *DistributionService* - una dintre componentele de bază ale arhitecturii. Acesta monitorizează QoS a sistemului şi oferă în timpul rulării implementarea corespunzătoare pentru un anumit serviciu local. DistributionService nu este utilizat direct de către aplicaţii. DistributionService comunică cu DistributionService din cloud, în scopul de a gestiona ciclul de viaţă al Riders.
- *Rider* - este utilizat de către DistributionService în cadrul procesului de migrare al Serviciului local în cloud. Rider conţine datele de funcţionare necesare pentru executarea şi instalarea serviciilor locale. Odată ce Rider instalează serviciul în cloud, el va return un smart-proxy la serviciul din cloud. Proxy-ul inteligent va fi utilizat de către aplicaţie ca și un inlocuitor pentru serviciul local. Rider serveşte drept recipient de migrare a datelor şi de codului şi este utilizat pentru a sprijini mobilitatea aplicațiilor.

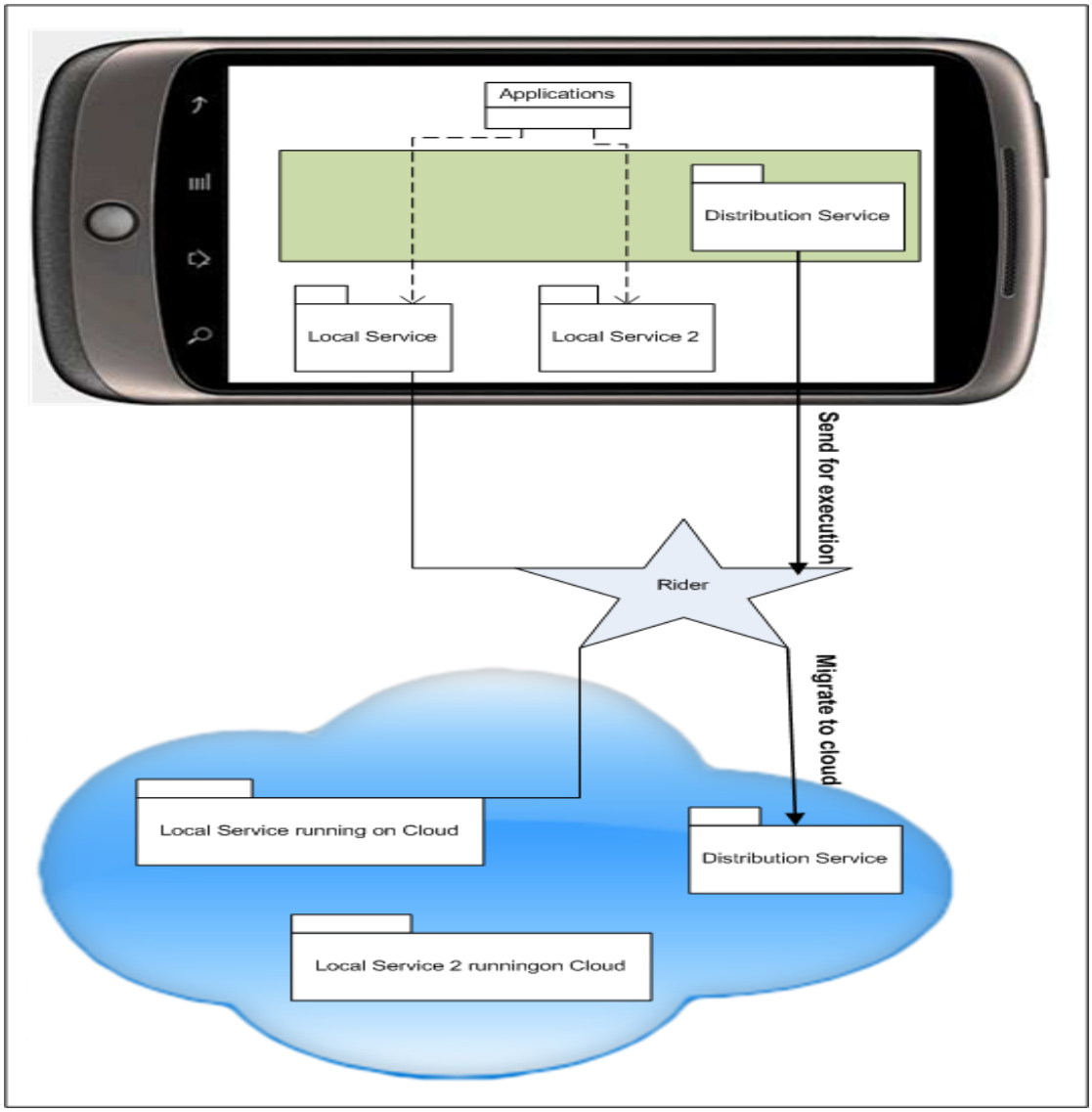

Figure 5.1 - Mobile cloudable framework architecture

DistributionService şi Rider susţin mobilităţii şi elasticitatea aplicaţiilor. DistributionService monitorizeaza QoS sistemului şi delegă apelurile la Serviciul local situat pe cloud, dacă este cazul.

DistributionService intercepteaz apelurile la serviciul local. Din această perspectivă, arhitectura propusă tratează cererea de distribuţie prin cloud ca o preocupare transversală. Distribuţia în cloud este separată de dezvoltarea aplicaţiei mobile. Cel mai bun mod de punere în aplicare astfel de arhitectura este de a utiliza Paradigma inovatoare de Aspect Oriented Programing.

5.3. Observații finale

Mobilitatea aplicației este una din caracteristicile cheie ale oricărei aplicaţii pervasive. O aplicație pervasive trebuie să fie capabilă să ruleze în medii diferite şi în plus, trebuie să fie distribuită fără probleme în medii diferite, fără intervenţia utilizatorului. Cercetarea actuală propune o soluţie pentru migrarea aplicațiilor mobile în cloud fără intervenţia utilizatorului, într-un mod care este transparent pentru dezvoltatorul de aplicaţii.

Resursele disponibile în cadrul cloud sunt incomparabile ca performanţă cu resursele disponibile pe dispozitivele mobile. Arhitectura descrisă în studiul actual a propus o soluţie inovativă pentru executarea sarcinilor de utilizator în cloud în cazul în care resursele locale nu sunt suficiente pentru a oferi utilizatorului o experienţă bună. Soluţia propusă are în vedere rularea în cloud a aplicației care a fost dezvoltată special pentru dispozitivele mobile.

Arhitectura propusă oferă detalii pentru o soluție inovatoare de realizare mobilității datelor și aplicațiilor. Arhitectura care propune utilizarea de sisteme cloud pentru executarea taskurilor mobile, s-a dovedit a fi viabilă atât de tehnic şi din punct de vedere al performanţei. Aplicaţie mobilă cloudable este în mod clar o soluţie viabilă pentru distribuirea de servicii pentru aplicaţii în cloud într-un mod care este transparent pentru dezvoltator de aplicaţii. Soluţia propusă nu forţează dezvoltatorul de aplicaţii în a utiliza un cadru particularizat, acesta este deschis şi se integrează perfect într-o aplicaţie mobilă. Dezvoltatorii au nevoie să se concentreze pe rezolvarea funcționalităților şi nu pe modul în care aplicaţia este distribuită în nor. Această sarcină este implementată de framework-ul propus.

Rezultatul acestui capitol este publicat în (Presecan S., 2011)

6. Securitate

Rețetele pervasive prevăd comunicarea între dispozitive informatice integrate în întreg mediul nostru. Acest lucru va duce la creşteri uriaşe în complexitatea infrastructurilor de reţea şi a serviciilor de informaţii disponibile în cadrul acestor infrastructuri. Prin urmare, provocarea de a gestiona serviciile de informaţii în acelaşi timp cu menţinerea securităţii şi a vieţii private va fi destul de ridicată.

Aceleași caracteristici fac sistemele pervasive convenabile și puternice, le face vulnerabile la noi atacuri de securitate şi ameninţări de confidenţialitate. Securitatea tradițională, mecanismele şi politicile existente nu pot să furnizeze garanţii corespunzătoare în relaţiile cu noile expuneri şi vulnerabilităţi.

În capitolul actual se identifică obstacolele și provocările care trebuie să fie depăşite de middleware pervasive, din perspectiva securității. Sectiunea finala a prezentului capitol prezintă o pontețială soluţie pentru tratarea problemelor de securitate într-un mediu pervasive.

Securitatea este o caracteristică-cheie pentru un sistem pervasive în care dispozitivele / utilizatorii se deplasează continuu printr-un mediu eterogen. Un middleware care ar putea fi folosit ca suport pentru implementarea sistemelor pervasive ar trebui să facă faţă provocărilor de securitate: de confidenţialitate, integritate, disponibilitate, autentificare, transparenţă, context-conştientizare, interoperabilitatea, prospeţime, ghidul de mobilitate, single sign-on. Unele dintre aceste provocări sunt specifice pentru sistemele pervasive, altele sunt provocările generale de securitate.

6.1. Detalii de proiectare

Este esenţial ca middleware utilizat pentru implementarea sistemului într-un mediu pervasive, să susţină diferite strategiile de securitate. Prin urmare, aceasta permite dezvoltatorilor de aplicaţii să utiliza unele dintre aceste strategii bazate pe nevoile de securitate ale aplicației vizate.
Cele mai importante clase care ar putea fi folosite pentru a defini un framework de securitate necesar pentru implementarea a unui middleware care suporta aplicatii pervasive, sunt prezentate în Figura 6.1 :

- *Security Manager:* aceasta este clasa principală utilizată pentru a efectua cele mai importante operaţiuni de *securitate:*. *Autentificare şi autorizare* SecurityManager ar putea fi utilizat direct de către aplicaţie în scopul de a valida accesul utilizatorilor sau indirect, de către interceptorii de securitate care ar putea fi aplicate în procesul de executare a diferitelor funcționalități. SecurityManager în sine este un observator al schimbărilor contextului. SecurityManager conţine una sau mai multe *SecurityStrategies* care ar putea fi utilizate pentru a efectua operaţiunile bazate pe diferite tipuri de configuraţii.
- *SecurityStrategy:* conţine configuraţii specifice care ar putea fi utilizate în scopul de a declanşa comportamente diferite în ceea ce securitatea. Fiecare SecurityStrategy conţine *SecurityProviders* diferite care sunt responsabile cu executarea operaţiunilor de securitate. Bazat pe configuraţia lor şi pe baza informaţiilor contextuale, a SecurityStrategy decide dacă s-ar putea face faţă cererilor de securitate sau nu, şi care a furnizorilor de securitate trebuie utilizate. SecurityStrategy conţine logica necesare pentru a decide dacă informaţiile utilizatorului sunt exacte sau nu. SecurityStrategies sunt în măsură să decidă în cazul în care utilizatorul are nevoie de a re-autentifice sau cerea de autentificare este încă valabilă.
- *SecurityProvider:* este responsabil pentru implementarea unui mod specific de manipulare a operaţiunilor de securitate. The SecurityProviders ar putea fi partajate între aplicaţii şi sunt capabili de a efectua operaţiunile de securitate prin utilizarea unui protocol specific / logica. De exemplu: ar putea efectua autentificare / autorizare prin intermediul unui WebService extern. În mod implicit, middleware ar putea oferi SecurityProviders pentru protocoale deschise: *OpenId* sau *OAuth.* Aceşti furnizori ar putea fi utilizați în scopul de a implementa operaţiunile de securitate pentru diferite sisteme care folosesc aceste standarde.

Un *SecurityProvider special* este *AccountManager. AccountManager* ar putea fi utilizat pentru a stoca token-uri de acces şi jetoanele de autorizare care ar putea fi împărţite între mai multe aplicaţii. Prin folosirea *AccountManager* utilizatorii trebuie să autentifice o singură dată pentru un anumit sistem şi apoi să poată folosi diferite aplicaţii pentru a se conecta la serviciile din sistemul ţintă. Astfel, dezideratul "single

sign-on" ar putea fi cu uşurinţă fi sprijinite de middleware pervasive. The *AccountManager* acţionează ca un mediu de stocare comun care stochează securizat informații de autentificare / autorizare.

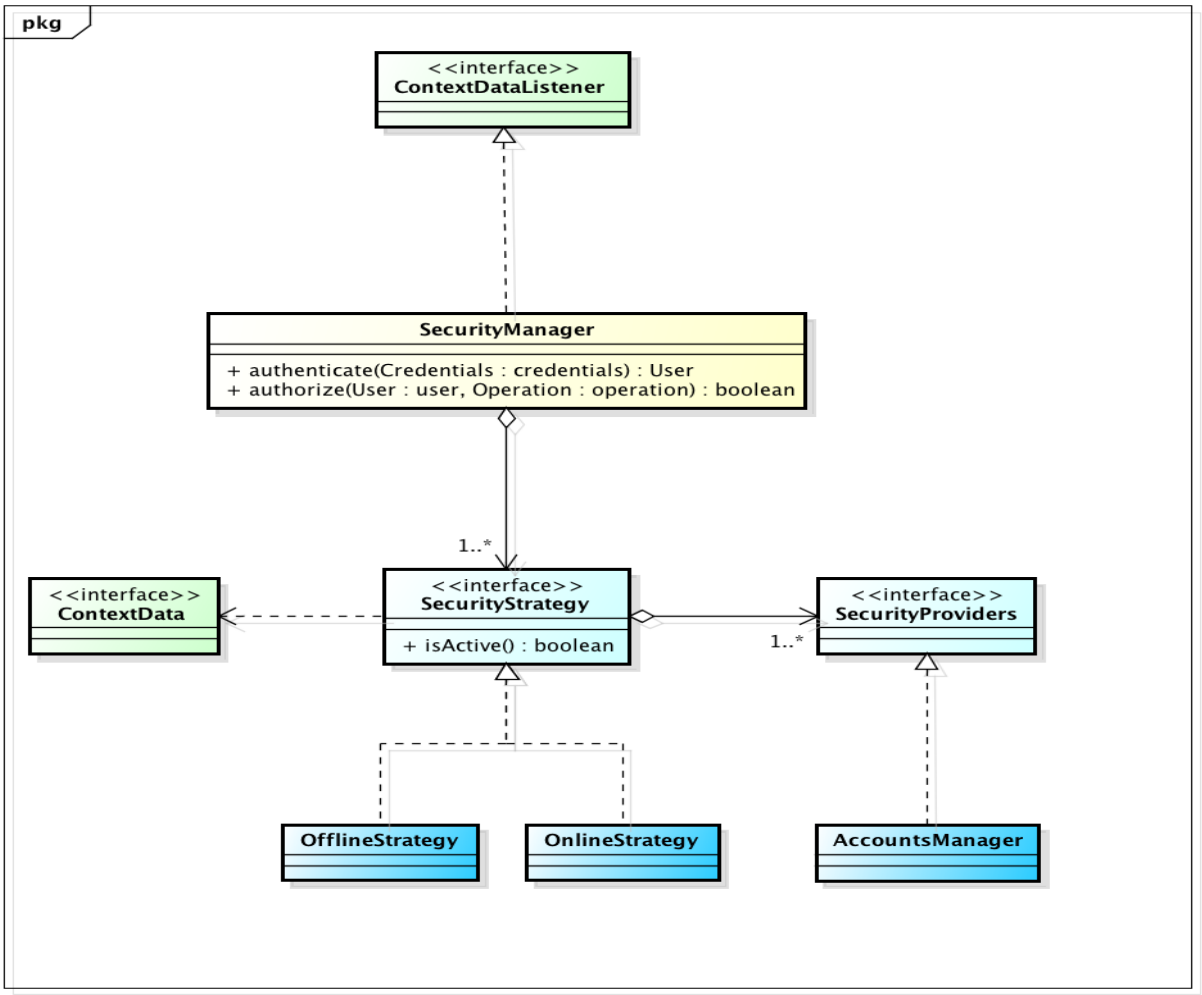

Figure 6.1 - Security framework design

6.2. Observații finale

Soluţia propusă are ca scop rezolvarea provocărilor de securitate care sunt specifice sistemelor de sensibile la context. Designul propus sprijină dezvoltarea strategiilor de securitate diferite, care ar putea fi puse în aplicare de către diferite aplicaţii sensibile la context. Soluţia propusă nu se referă la criptare a datelor sau semnarea de date. Ele ar trebui să fie suportate de către middleware care asigură transportul de date şi integritatea datelor.

Soluţia propusă se potriveste bine cu middleware conştiente de context definite în Capitolul 3. Soluţia curentă va fi extinsă şi integrată în scopul de a oferi un middleware complet, care sprijină toate provocările identificate pentru un sistem pervasive.

Rezultatele a capitolului curent au fost parţial publicate în (Presecan S., 2008)

7. Arhitectura unui middleware pervasive

Capitolele anterioare au prezentat diferite detalii de proiectare ale unor micro-sisteme utilizate pentru rezolvarea provocările specifice a sistemelor pervasive. Prezentul capitol cuprinde un studiu de caz care are drept scop identificarea principalelor provocări pentru un sistem pervasive utilizat pentru rezolvarea zi de zi probleme într-o comunitate de studenţi. Odată ce sunt identificate problemele, o solutie arhitecturala este propusă în vederea soluţionării provocările identificate.

Soluţia arhitecturală propusă agregă soluțiile propuse în capitolele anterioare. Rezultatul este o arhitectură generică care ar putea fi utilizată pentru implementarea de soluții informatice în diferite domenii de afaceri în care utilizarea puterii de calcul este omniprezentă.

7.1. Viziune

Aproape toate universităţile au pus în aplicare în prezent, portaluri educaţionale, de unde studenții pot obţine informaţii relevante. Cele mai multe dintre aceste portaluri sunt de tip web-centric, expunând informaţii pe web. În multe cazuri, acest lucru este suficient, dar există îmbunătăţiri care pot fi făcute în scopul de a facilita colaborarea între studenţi şi universitate. Aproape toate e-portaluri suportă o interacțiune unidirecțională. Studenţii pot accesa direct informaţiile publicate pe portal educaţional, prin intermediul interfeţei web.

Având în vedere progresele actuale în lumea de hardware şi software, aceste opţiuni nu sunt suficiente. In zilele noastre, telefoanele inteligente şi reţelele sociale sunt omniprezente. Studenții doresc să fie informaţi pretutindeni şi doresc să facă schimb de informaţii cu colegii lor atât în interiorul campusului cât și când sunt mobili. Utilizarea reţelelor sociale este, uneori, mult mai uşor decât schimbul de informaţii prin intermediul canalelor tradiţionale, cum ar fi e-mail şi telefon.
Sistemul informatic ar trebui să fie, de asemenea, conştient despre profilul studenţilor. Pe baza acestor profiluri, studenții putând să primească anumite mesaje personalizate sau să aibă acces la anumite informaţii destinate lor.

Studenţii şi profesorii ar trebui să fie legaţi printr-o reţea virtuală, care este conştientă de preferinţele şi profilurile lor, facilitând de asemenea schimbul de informaţii de la orice dispozitiv, din orice locaţie.

7.2. Provocări

Pornind de la viziune mai sus enunţată şi având în vedere provocările universale descrise la paragraful 2.2, am indentificat ca următoarele provocări ar trebui să fie luate în considerare de către orice dezvoltator care are ca scop implementarea unui sistem pervasive care are caracteristicile descrise mai sus:

- *Context-awareness:* sistemul trebuie să fie conştient de profilul/contextul studentului: profil geografic, profilul social, context de mediu, context temporal.
- *Invizibilitate:* sistemul ar trebui să reacţioneze la schimbările contextului. De fiecare dată când contextul se schimbă, sistemul este conştient cu privire la modificare şi încearcă să se adapteze funcţionalităţilor sistemului la noul context fără să îl perturbe pe utilizator.
- *Mobilitatea aplicaţiei:* Pe aproape toate portalurile educaţionale existente, datele sunt transferate utilizatorului. Datele sunt mobile, dar aplicaţiile nu. Există, de asemenea, o nevoie pentru a muta aplicaţiile între dispozitive. Utilizatorii se aşteaptă să-şi continue munca lor pe dispozitive diferite.
- *Securitate:* nevoia de mobilitate şi aplicaţiile conştiente de context sunt expuse la atacuri diferite. Utilizatorii doresc să efectueze toate operaţiunile descrise mai sus într-un mod securizat.
- *Integrare:* Un portal educaţional omniprezentă este un sistem eterogen, care integrează diferite tipuri de sisteme de calcul: servere pentru stocarea de date şi pentru executarea sarcinilor funcţionale, laptopuri şi computere personale pentru a accesa informaţiile din locuri fixe, telefoane inteligente pentru acces mobil. Toate acestea trebuie sa fie integrate unitar în sistem
- *Scalabilitate locală:* numărul de dispozitive conectate la portal educaţional omniprezent poate varia în mod substanţial în funcţie de locaţie şi de oră. Sunt, de asemenea, perioade când sistemul se află sub stres de utilizare.

Sistemul ar trebui să suporte scalabilitate locală pentru a face față acestor fluctuații de access.

- *Implementare și instalare:* middleware sistemul ar trebui să ajute dezvoltatorii să implementeze aplicații care sunt capabile de a utiliza informațiile primite din surse diferite, folosind protocoale diferite. Middleware ar trebui să faciliteze dezvoltarea și implementarea aplicațiilor pe diferite tipuri de dispozitive care ruleaza sisteme de operare diferite. Middleware ar trebui să furnizeze, de asemenea, servicii și funcționalități care ajută dezvoltatorii de aplicatii pentru reducerea costurilor generale de dezvoltarea de aplicații.

7.3. Arhitectură Middleware

Pornind de la viziune și având în vedere provocările identificate mai sus, următoarea arhitectură este propusă pentru implementarea unui sistem pervasive care ar putea fi utilizate pentru realizarea următoarelor versiuni de portaluri educaționale. Arhitectura combină toate modelele prezentate în capitolele anterioare. Soluția își propune să rezolve toate provocările indentificate printr-un middleware generic folosit pentru sistemele pervasive.

Arhitectura middleware prezentată în figura 7.1, are următoarele componente principale:

- ContextService
- DistributionService
- SecurityManager
- CommunicationChannel

7.3.1. ContextService

ContextService este componenta utilizată pentru a oferi suport pentru funcționalități legate de context-awareness și invizibilitate. *ContextService* observă mediul înconjurător prin folosirea de senzori și generatori. Senzorii ar putea fi încorporați în smartphone-uri, cum ar fi senzorii GPS, sau senzori externi. Prin intermediul senzorilor ContextService este notificat ori de câte ori informațiile colectate de către acești senzori sunt în schimbare. ContextService este, de asemenea, conectat la

generatoare care să furnizează informaţii la cerere cu privire la diferite contexte, cum ar fi: contextul social, contextul funcţional, contextul temporal.

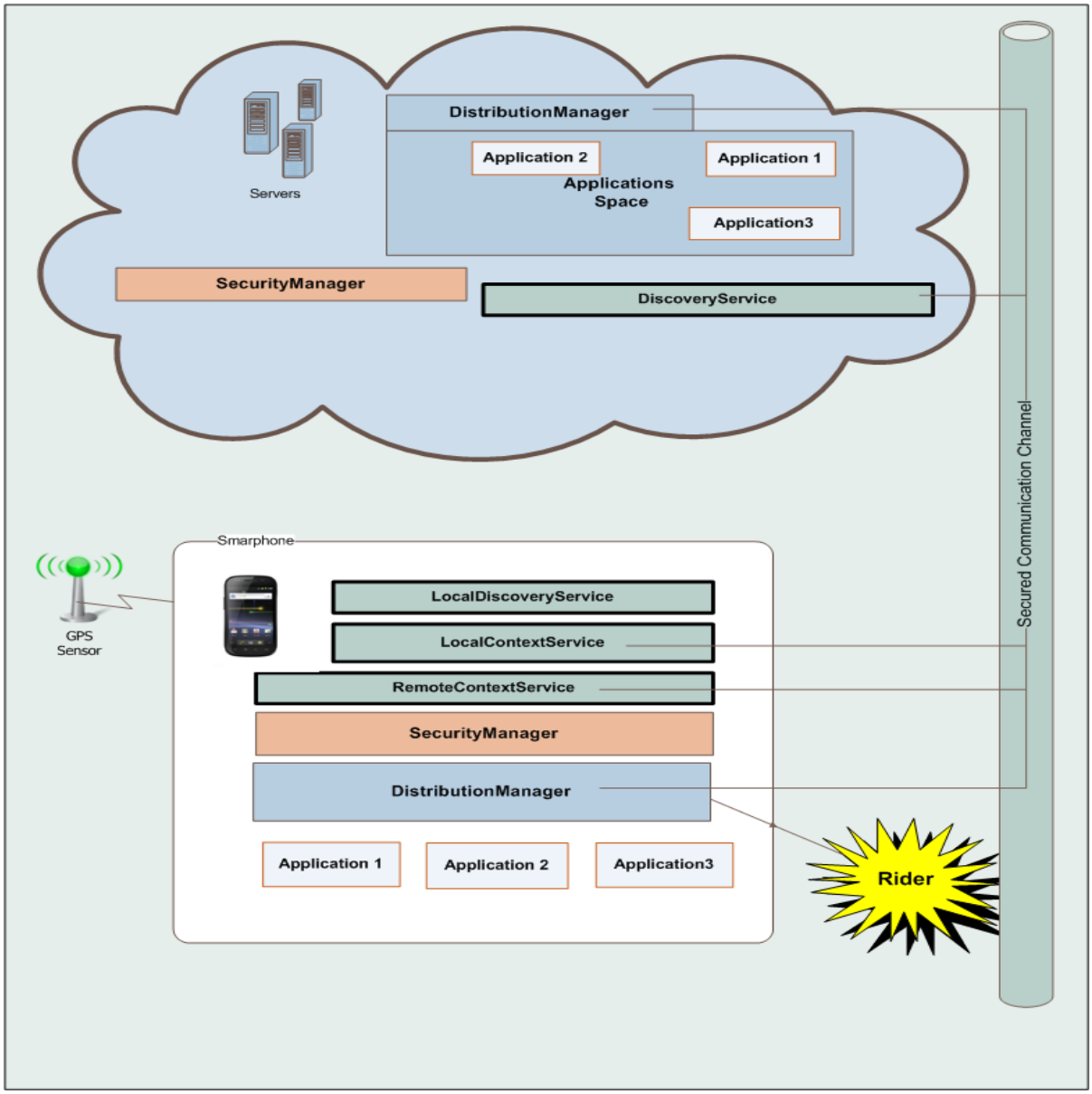

Figure 7.1 - Middleware architecture

7.3.2. DistributionService

Utilizatorii se aşteaptă să își continue activităţile lor, prin utilizarea unor medii cu capacități de calcul diferite. Ei doresc să aibă aplicațiile și datele migrate de la un dispozitiv la altul şi de asemenea, ei vor să aibă o utilizare cât mai eficientă a puterii de calcul existente într-un anumit sistem. Middleware este responsabil de migrarea

datelor şi aplicațiilor corespunzătoare între diferite dispositive, în vederea îndeplinirii sarcinilor de utilizator. Mobilitatea aplicației este transparentă pentru utilizator.
DistributionService efectuează de distribuţia aplicațiilor împreună cu datele lor şi cu context în care userul se află. DistributionService ştie cum să găsească un loc mai bun pentru executa anumite sarcini primite de la utilizator, prin monitorizarea parametrilor QoS a diferitelor dispozitive şi servere conectate la sistem.

7.3.3. Managerul de securitate

SecurityManager efectuează toate operaţiunile de necesare pentru a sprijini: autentificarea, autorizarea şi confidenţialitatea datelor. SecurityManager utilizează diferite strategii pentru a autoriza accesul utilizatorului. Într-un sistem pervasive , securitatea poate fi aplicată transversal şi în mod transparent la operaţiunile execute de către utilizatori. SecurityManager sprijină implementarea securității prin furnizarea suportului pentru următoarele operaţiuni: autorizare şi autentificare.

7.3.4. Canalul de comunicare

Un portal educaţional care este pregatit să aibă o utilizare omniprezenta trebuie să integreze diferite tipuri de dispozitive, senzori, generatoare de context şi calculatoare. Fiecare dintre ele utilizează protocoale diferite şi diferite canale de comunicare. Eforturile de integrare fără a definit un canal de comunicare clar, este un coşmar. Canalele de comunicare necesare pentru a integra diferite aplicații care rulează pe dispozitive diferite, sunt o provocare majoră pentru sistemele pervasive.

Canalul de comunicare este capabil să ofere siguranţă datelor care sunt preluate la către clienţi sistem extern sau de către diferite dispozitive care sunt conectate in sistem. Interfaţa de interogare a datelor poate fi expusă ca şi webservices RESTfull prin HTTP / HTTPS. Prin servicii web RESTfull datele ar putea fi citite / actualizate / şterse de către dispozitivele conectate la middleware.

Trimiterea în schimb a notificărilor despre modificările de context peste HTTP nu este la fel de simplu de realizat. În mod implicit, HTTP este un canal de date unidirecțional. Există însă mai multe tehnologii care au rezolvat această problemă.

Una din soluţii este de a utilizarea conexiunile HTTP persistente şi de a folosi aceste conexiuni pentru a trimite mesajele bidirecţional. Un astfel de exemplu este **Bidirectional-streams Over Synchronous HTTP (BOSH)**. BOSH este proiectat pentru a transporta orice fel de date eficient şi cu latenţă minimă în ambele direcţii. Poate fi utilizat de către aplicaţii care necesită atât "push" şi "pull" pentru date. BOSH foloseşte mai eficient lăţimea de bandă decât majoritatea celorlalte protocoale bidirecţional de transport bazate pe HTTP. (Paterson, Smith, Saint-Andre, şi Moffitt, **2010).**

Tehnologia BOSH realizează atât latenţă redusă şi consum redus de lăţime de bandă, prin încurajarea Connection Manager pentru a nu răspunde la o cerere până când există date pentru a fi trimise la client. De îndată ce clientul primeşte un răspuns de la ConnectionManager el poate trimite o altă cerere, astfel asigurându-se că ConnectionManager deţine întotdeauna un canal activ de comunicare pe care il poate folosi pentru a "împinge" date la client.

Figure 7.2 prezintă modalităţile de a interacţiona cu canalele de comunicare:

- ContextService expune datele prin folosirea de QueryInterface. QueryInterface este definit ca un serviciu web RESTfull care se realizează de peste HTTPS
- QueryInterface este capabil să înregistreze cererile de notificare prin utilizarea tehnicilor de notificari peste HTTP. Notificările sunt distribuite peste HTTP folosind protocolul BOSH. Prin folosirea protocolul BOSH aplicațiile ar putea fi notificate atunci când anumite valori context sunt schimbate.
- Autoritatea de Certificare ar putea fi utilizată de către anumite aplicaţii pentru a elibera certificate, care ar putea fi folosite apoi pentru a semna informaţii sensibile
- Aplicațiile / utilizatorii sunt conectaţi peste HTTPS la serviciile de RESTfull, în scopul de a obţine securizat informaţiile expuse de către diferite servicii publice

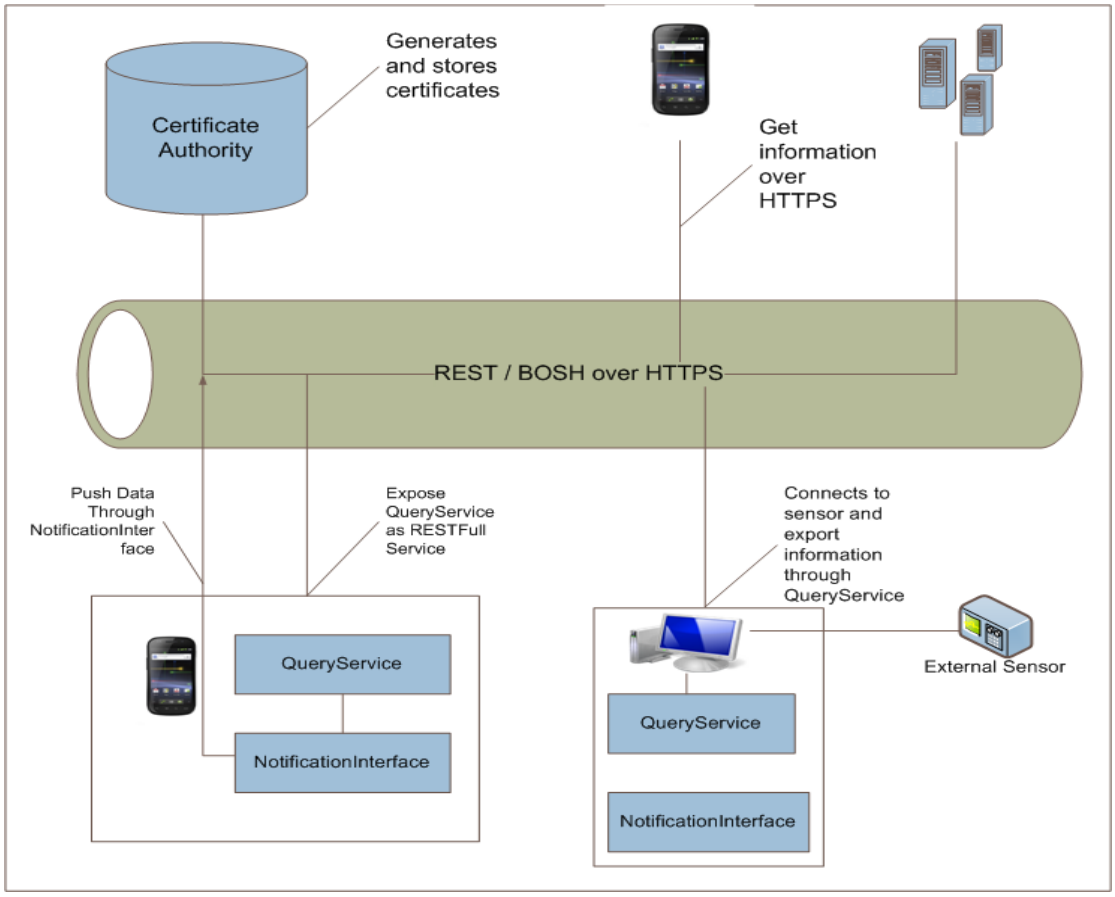

Figure 7.2 – Canalul de comunicare

Scopul oricărui middleware este de a sprijini implementarea a funcționalităților sistemului şi pentru a uşura efortul dezvoltatorilor. Având un canal de comunicare care se bazează pe HTTP şi BOSH, integrarea este mult facilitată.

7.4. Observații finale

În prezent, calculatorul personal dispare, el devine un instrument care este utilizat de către utilizatori pentru a efectua anumite sarcini. În prezent, accentul se pune pe informaţii şi pe aplicaţiile şi sistemele care facilitează accesul la informaţii. Portalul educațional omniprezent este o sarcină dificilă, care începe mai întâi cu o viziune pe care am detaliat-o în acest capitol. Pornind de la viziune am identificat provocările care trebuie să fie rezolvate în vederea implementării a unui sistem de educațional omniprezentă.

Pornind de la viziune şi de la provocările, am propus o arhitectura pentru middleware, care ar putea fi utilizat pentru implementarea a unui portal educaţional omniprezentă.

Arhitectura propusă combină toate componentele descrise în capitolele anterioare, şi prevede modalităţi de rezolvare coerentă a provocărilor identificate. Middleware propus este generic, prin urmare, ar putea fi utilizat pentru implementarea a oricărui tip de sistem pervasive.

Arhitectura propusă rezolvă unele dintre cele mai importante provocări extinse: context-awarenes, invizibilitatea, integrare, mobilitatea aplicațiilor și a datelor, securitate şi facilitarea implementării. Middleware propus se bazează pe standarde deschise şi să nu forțează dezvoltatorii să utilizeze tehnologii proprietar. Standardele propuse au un suport mare în comunitate dezvoltatorilor, ceea ce va diminua costurile totale ale soluţiei. Protocoalele de comunicare propuse (HTTP şi BOSH) permit obținerea da informații prin interogări cât și primirea de notificări referitoare la modificările de context. Astfel middleware-ul propus poate fi folosit pentru implementarea oricărui sistem pervasive care are ca și scop rezolvarea provocărilor indentificate la începutul capitolului.

8. Concluzii

Primele capitole ale tezei se concentrează pe înţelegerea provocărilor sistemelor pervasive. Am identificat, astfel, provocările-cheie pentru un sistem pervasive ca fiind: context-awareness, invizibilitatea, servintegrare, securitate, mobilitate aplicațiilor și a datelor, ușurința de dezvoltare şi implementare. Aceste concepte-cheie sunt prezentate în detaliu în capitolul 2. Al doilea capitol, de asemenea, se axează pe prezentarea statutul celor mai recente cercetări în domeniul pervasive middleware şi a modului în care cercetarea teoretică poate oferi soluţii pentru implementare sistemelor care ar putea rezolva problemele enumerate mai sus. Al doilea capitol nu se pretinde a fi o analiză completă a tuturor pervasive middleware existente. Cu toate acestea, am încercat să analizăm cele mai importante pervasive middleware. Middleware existente, sunt analizate din perspectiva modul în care a depăşesc provocările identificate. Prin analizarea acestora, am ajuns la concluzia că aceste middleware reprezintă de fapt un bun punct de plecare pentru colectarea de idei despre modul cum să arate un middleware ideal.

Al doilea obiectiv al cercetării de faţă este conceperea unui middleware pervasive şi ne-am concentrat pe îndeplinirea acestui obiectiv, începând cu al treilea capitol. Contextul-awareness este un concept cheie în pervasive computing, este unul dintre diferenţiatori între mobile și pervasive computing. Strădania noastră de a crea un middleware pervasive a început cu proiectarea unei componente a middleware care are ca scop pentru a rezolva provocările context-awareness. Designul propus transmite ideea de a avea un "buddy" ContextServices, care ar putea colabora pentru furnizarea de informaţii despre mediul înconjurător. Modul de distribuţie a acestor "buddies" este transparent pentru dezvoltator de aplicaţii şi middleware este capabil de a descoperi şi conecta acești "buddies" împreună. DiscoveryService care monitorizează QoS, ar putea înlocui "buddies" ContextProviders în timpul rulării. Această înlocuire dinamică asigură, de asemenea, o scalabilitate locală, care reprezintă o provocare majoră pentru un middleware pervasive.

Împreună cu Context-awarenes, Invizibilitatea este o caracteristică cheie al oricărui sistem pervasive. Al patrulea capitol oferă detalii despre provocările Invizibilității şi despre posibilităţile de a rezolvare a acestora. Middleware propus conţine configuraţii pentru reacţiile sistemului la schimbările contextului. RuleManager este un ContextListener, prin urmare, aceasta este notificat ori de câte ori valorile de context monitorizate sunt modificate. RuleManager ar putea fi configurat static sau dinamic cu reguli de reacţie. Astfel, poate învăța care este comportamentul utilizatorului şi ținând cont de preferinţele ar putea reacţiona la schimbările contextul în numele utilizatorului. Adaptare sistem, fără intervenţia utilizatorului este o caracteristica-cheie care sprijină asigurarea invizibilității.

Mobilitatea Aplicaţiilor este o caracteristică diferenţiatoare între sisteme distribuite şi cele pervasive. Este o provocare destul de dificil de realizat, având în vedere eterogenitatea de dispozitivelor conectate într-un sistem pervasive. În capitolul al cincilea, ne-am propus să dezvoltam un mod inovator de a sprijini cererea de migraţie a aplicațiilor de la dispozitive inteligente la alte dispozitive sau chiar în cloud. Prin folosirea DistributionService, Rider şi ClassLoader personalizate, servicii aplicațiilor ar putea migra în cloud sau de la orice dispozitiv din apropiere mai puternic, pentru a avea acces la o putere de calcul mai mare pentru executarea sarcinilor funcționale. Mobilitatea aplicațiilor este susţinută de middleware şi este pusă în aplicare în mod transparent pentru dezvoltator de aplicaţii. Prin folosirea paradigmei AOP, distribuţia în cloud este implementat ca și un *cross-cutting concern*. Distribuţie în cloud ar putea fi decisă în mod automat de către middleware prin monitorizarea parametrilor QoS a serviciilor. Pentru utilizatorul final, distribuţia de servicii în cloud se face într-un mod transparent.

Capitolul 6 oferă informaţii detaliate cu privire la implicaţiile securității în pervasive computing. Securitatea este percepută ca un *cross-cutting concern*, astfel de autorizarea ar putea fi implementată prin utilizarea *advices* AOP.

Arhitectura middleware-ului pervasive este definită şi detaliată în capitolul 7. Arhitectura combină toate modelele prezentate în capitolele anterioare şi propune o middleware generic care ar putea fi folosit pentru implementarea oricărui sistem

pervasive. Middleware vizează depăşirea provocărilor definite pentru un sistem educațional pervasive. Middleware sprijină context-awarenes, invizibilitatea, securitate, mobilitate aplicaților, scalabilitate locală şi integrare. Merită menţionată modalitate inovatoare de a notifica modificările valorilor de context, dispozitive / sisteme care sunt interesaţi de acest valori. Utilizarea BOSH peste protocolul HTTP simplifică trimiterea acestor notificări prin folosirea unor protocoale standard. Evenimentele sunt astfel distribuite la abonaţi prin utilizarea de transport HTTP. Middleware propus se bazează pe standarde deschise, cum ar fi: REST, BOSH, HTTP (s) şi, prin urmare, nu va bloca dezvoltatorii forţându-i să utilizeze un anumit framework proprietar. Utilizarea de standarde deschise sprijină şi facilitează integrarea diferitelor tipuri de dispozitive care rulează pe sisteme de operare diferite.

Comparând middleware propus cu la sistemele analizate: AURA, Gaia, One.World, Pico si SODA este evident că are o mai bună acoperire în ceea ce priveşte sprijinirea diferitelor caracteristici specific pervasive computing. Sistemele analizate sunt în măsură să îndeplinească numai anumite provocări, ele nu au fost create cu scopul de a oferi o solutie end-to-end pentru sisteme pervasive.

Prin teza de faţă, ne-am propus să proiectăm o soluţie de state-of-the-art pentru un middleware care se concentrează să acopere toate aspectele majore ale pervasive computing. Soluția propusă se dorește a fi una completă şi credem că este una unică, din prisma propunerilor sale inovatoare, cum ar fi:

- buddy ContextServices, care sunt interconectate într-o reţea de furnizori şi care sunt înlocuite dinamic ori de câte ori unul dintre ei nu funcţionează bine sau este deconectat de la reţea
- distribuţia transparentă a sarcinilor de execuție în cloud pentru realizarea scalabilității locale şi a mobilităţii aplicațiilor
- folosirea AOP pentru implementarea a unei distribuţie aplicațiilor într-un mod implicit fără a forța dezvoltatorul să își modifice codul existent
- folosirea BOSH pest HTTP pentru distribuirea evenimentelor legate de modificarea contextului

Chiar dacă soluţia propusă a fost implementată şi testate doar în exemple de laborator şi, parțial, ca și parte dintr-un produs comercial de succes, cercetarea actuală trebuie să fie extinsă în scopul de a sprijini mai multe limbaje de programare şi de a rula pe multe platforme. Exemplele de laborator au fost implementate folosind Java și Android, dar în vederea demonstrării completitudinii sale, soluția propusă trebuie extinsă în continuare, cu scopul de a facilita integrarea şi dezvoltarea de aplicații utilizând diferite platforme.

9. Bibliografie

[1] Adams, A. (2003). A whole picture is worth a thousand words. *SIGCHI Bulleting (Supplement to Interactions)* , Volume 2003, May-June , pp. 121-124.

[2] Adams, A. (1999). Users' perception of privacy in multimedia communication. *Conference on Human factors in computing systems* (pp. 53-54). ACM Press.

[3] Adams, A., & Sasse, M. (1999). Privacy Issues in Ubiquitous Multimedia Environments: Wake Sleeping Dogs, or Let Them Lie?, INTERACT, pp. 214-221.

[4] Adelstein, F., Gupta, S., Richard, G., & Schwibert, L. (2005). *Fundamentals of Mobile and Pervasive Computing.* New-York: McGraw-Hill.

[5] Anthony, D., Joshua, M., & Tauber, A. (1997). Mobile computing with the rover toolkit. *IEEE Transactions on Computers*. Volume 46, pp. 337 – 352

[6] Appweb. (2007, 01). *AppWeb Server*. Retrieved 08 2011 from AppWeb Server: http://appwebserver.org/

[7] Aura, P. (2002). *Project Aura*. Retrieved 07 2008 from http://www.cs.cmu.edu/~aura/

[8] Bapat, S. (1994). Object-Oriented Networks, Models for Architecture, Operations and Management. New-York: Prentice-Hall International.

[9] Becker, C., & Schiele, G. (2003). Micro-broker- based middleware for pervasive computing, *Pervasive Computing and Communications (PerCom),* pp. 443 - 451

[10] Bellotti, V. (1997). Design for privacy in multimedia computing and communication environments. Massachusetts: MIT Press.

[11] Bellotti, V., & Sellen., A. (1993), Design for Privacy in Ubiquitous Computing Environments, *European Conference on computer supported cooperative work*, pp. 77-92

[12] Birrell, A., & BJ, B. N. (1994). Implementing remote procedure calls. *ACM Trans Computer Systems ,* Volume 2 (1), pp. 39-59.

[13] Brandram, J., & Hansen, T. (2004). The AWARE architecture: supporting context-mediated social awareness in mobile cooperation, *Proceedings of the*

2004 ACM conference on Computer supported cooperative work, pp. 192 – 201

[14] Campbell Roy, Al-Muhtadi J., Naldurg P., Sampemane G. and Dennis M. (2003), Towards Security and Privacy for Pervasive Computing, *Software Security — Theories and Systems,* Volume 2609/2003, p. 77-82

[15] Carriero, N., & Gelernter, D. (1986). *The s/net's linda kernel.* ACM Trans Comput Syst .

[16] Chan, E., & Bresler, J. (2005). Gaia Microserver: An Extendable Mobile Middleware Platform. *Third International Conference on Pervasive Computing and Communication*, Santa Cruz: IEEE, pp. 309-313

[17] Chappell, D. (2006). *Understanding .NET (2nd Edition).* Addison-Wesley Professional .

[18] Cox, P. A. (2011). *developerWorks*. From Mobile cloud computing: http://www.ibm.com/developerworks/cloud/library/cl-mobilecloudcomputing/

[19] Crane, D., & McCarthy, P. (2008). Comet and Reverse Ajax: The Next-Generation Ajax 2.0. Apress.

[20] Dey, A., & Abowd, G. (1999). *Towards a better understanding of context and context-awareness.* Georgia: Georgia Institute of Technology.

[21] Doceur J. (2002), The Sybil Attack, *Peer-to-Peer Systems,* Volume 2429/2002, pp. 251-26

[22] Edu, Berkely (n.d.). *Endeavour System*. Retrieved 07 2008 from http://endeavour.cs.berkeley.edu

[23] Edu, M. (2003). From Oxygen System: http://oxygen.lcs.mit.edu/

[24] Eugster, P., Felber, P., & Guerraoui, R. (2003). *The many faces of publish/subscribe.* ACM Computer Surveys, Volume 3, pp. 114 - 131

[25] Fielding, R. (2000). *Phd. Disertation Architectural Styles and the design of Network-based Software Architecture*. Retrieved 2008 from http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[26] Forum, U. (2008). *Universal plug and play device architecture*. From UPnP: . http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf

[27] Garlan, D., Siewiorek, D., & Smailagic, A. (2002). Project Aura: Toward Distraction-Free Pervasive Computing. *Pervasive Computing*, Volume 1(2), pp. 22 - 31

[28] Google. (2010). *Robo Guice*. From http://code.google.com/p/roboguice/

[29] Google. (2011, 04 29). *The Mobile Movement: Understanding Smartphone Users*. Retrieved 04 2011 from http://googlemobileads.blogspot.com/2011/04/complimentary-copy-of-mobile-movement.html

[30] Google. (2008). *What is Android*. From http://developer.android.com/guide/basics/what-is-android.html

[31] Google, A. (2009, 12). *Trip Journal winner of Android Development Challenge Organized by Google*. From Android Development Challenge: http://code.google.com/android/adc/gallery_travel.html

[32] Gravelle, R. (2008, 03). *Webreference*. From Comet Programming: Using Ajax to Simulate Server Push: http://www.webreference.com/programming/javascript/rg28/

[33] Grimm, R. (2004). One.world: experiences with a pervasive computing architecture. *Pervasive Computing*, Volume 3, pp. 22 - 30

[34] Grimm, R., Anderson, T., Bershad, B., & Wetherall, D. (2004). System Support for Pervasive Applications. *Trans. Computer Systems , 22* (4), -.

[35] Group, O. M. (2008). *Common object request broker architecture (corba/iiop)*. From OMG: http://www.omg.org/spec/CORBA/3.1/

[36] Hyun Jung La, S. D. (2010). A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services. *3rd International Conference on Cloud Computing.* Miami: IEEE, pp. 466 - 473

[37] Interface 21, S. F. (2006). *Aspect Oriented Programming with Spring*. From http://static.springsource.org/spring/docs/2.0.x/reference/aop.html

[38] Ipina, D., & Katsiri, E. (2001). An ECA Rule-Matching Service for Simpler Development of Reactive Applications. *Published as a supplement to the Proc. Of Middleware,* IEEE,

[39] Jetty. (2003). *Jetty webserver*. Retrieved 08 2011 from Jetty webserver: http://jetty.codehaus.org/jetty/

[40] Kagal, L., Finin, T., & A., J. (2001). Trust-Based Security in Pervasive Computing Environments. *IEEE Computer* , Volume 34 Issue 12

[41] Knorr, E., & Gruman, G. (2010). *What cloud computing really means*. From Inforworld: http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031?page=0,1

[42] Koch, C. (2005). How SOA Really Works. *CIO*, http://www.cio.com/article/10591/How_SOA_Really_Works

[43] Krill, P. (2010, 10). *AJAX alliance recognizes mashups*. Retrieved 08 2011 from Infoworld: http://www.infoworld.com/d/developer-world/ajax-alliance-recognizes-mashups-559

[44] Kumar, M., Shirazi, B., & Singhal, M. (2003, July-Sept). PICO: A Middleware Framework for Pervasive Computing, *IEEE Pervasive Computing,* Volume 2, pp. 72 - 79

[45] Laerhoven, K. V. (1999). Online Adaptive Context Awareness, starting with low- level sensors. Free University of Brussel.

[46] Mangoose. (2008). *Mangoose*. Retrieved 08 2011 from Mangoose Web Server: http://code.google.com/p/mongoose/

[47] Manjunatha, A., Ranabahu, A., Sheth, A., & Thirunarayan, K. (2010). Power of Clouds In Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development. *Cloud Computing Technology and Science.* IEEE.

[48] Mansukhani, M. (2005). *Service Oriented Architecture.* Hewlett Packard .

[49] Mei, L., Chan, W., & Tse, T. (2008). A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues. *Asia-Pacific Service Computing.* IEEE, pp. 464 - 469

[50] Microsystems, S. (2006). *remote method invocation (rmi) - related apis and developer guides.* . From Java Website: http://java.sun.com/javase/6/docs/technotes/guides/rmi/index.html

[51] Mitchell, K. (2002). *Supporting the Development of Mobile Context-Aware Computing.* Department of Computing. Lancaster University.

[52] Monnox, A. (2005). Rapid J2EE™ Development: An Adaptive Foundation for Enterprise Applications. Prentice Hall.

[53] Newsome, J., & Shi, E. (2004). The Sybil attack in sensor networks: Analysis and Defense. *International Symposium on INformation Processing in Sensor Networks*, US: ACM Press, pp. 225-229

[54] Nokia. (2006). *Mobile Web Server*. Retrieved 2011 from Nokia Labs: http://research.nokia.com/page/231

[55] Norman, A. (1998). *The Invisible Computer.* Cambridge. Massachusetts: MIT Press.

[56] One.world. (2003, 02). *One.world*. Retrieved 07 2008 from http://cs.nyu.edu/rgrimm/one.world/

[57] Orlando, D. (2009, 01). *Cloud computing service models*. Retrieved 04 2011 from developerWorks: http://www.ibm.com/developerworks/cloud/library/cl-cloudservices1iaas/

[58] Oxygen, M. P. (2002). *Pervasive Human-Centred Computing*. Retrieved 12 2006 from MIT Laboratory for Computer Science: http://oxygen.lcs.mit.edu/Overview.html

[59] Paterson, I., Smith, D., Saint-Andre, P., & Moffitt, J. (2010, 07 02). *Bidirectional-streams Over Synchronous HTTP (BOSH)*. Retrieved 08 2011 from XMPP Standards Foundation: http://xmpp.org/extensions/xep-0124.html

[60] Ponnekanti, S., Johanson, B., Kiciman, E., & Fox, A. (2003). Portability, extensibility and robustness in iRos. *Pervasive Computing and Communications(PerCom 2003).* IEEE, pp. 11 - 19

[61] Presecan, S. (2009). Distributed context provisioning middleware. Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques Conference (KEPT2009), Section 'Knowledge Processing in Economics', Babes-Bolyai University of Cluj-Napoca

[62] Presecan, S. (2011). Mobile Cloudable Applications - New Way of Distributing Mobile Tasks Into the Cloud - TO BE PUBLISHED. *International Workshop on Machine-to-Machine Communications, IEEE GlobeCom.* Huston: IEEE.

[63] Presecan, S. (2009). Pervasive context-aware middleware. *Proc. 9th International Conference on Informatics in Economics.* Bucharest: ASE.

[64] Presecan, S. (2007). Pervasive Education Portal - Architectural Challenges. *Proc. 8th International Conference on Informatics in Economics* (pp. 108-113). Bucharest: ASE.

[65] Presecan, S. (2008). Security Challenges for Pervasive Computing. *The Romanian Workshop on Mobile Systems, Economy Informatics.* Cluj-Napoca: FSEGA.

[66] Presecan, S. (2008). SOA based architecture for pervasive systems. *The Romanian Workshop on Mobile Systems, Economy Informatics*, Cluj-Napoca: FSEGA.

[67] Presecan, S., & Tomai, N. (2009). Distributed Context Provisioning and Reaction Middleware. *Intelligent Computer Communication and Processing.* Cluj-Napoca: IEEE Computer Society Press, pp. 351 - 354

[68] Qian Wang, D. R. (2009). SOA's Last Mile-Connecting Smartphones to the Service Cloud. IEEE international conference on *Cloud Computing.* Bangalore : IEEE, pp. 80 - 87

[69] Ricky K., K. M.-L. (2010). A Stack-on-Demand Execution Model for Elastic Computing. *39th International Conference on Parallel Processing.* San Diego: IEEE, pp. 208 - 217

[70] Rivest, R., & Stanajo, F. (2002). *Security for ubiquitous computing.* US: Willey.

[71] Robinson P., Vogt H. and Wagealla W. (2005), Some Research Challenges in Pervasive Computing, Privacy, *Security and Trust within the Context of Pervasive Computing,* Volume 780, pp. 1-16

[72] Roman, M., & Campbell, R. (2000). GAIA: Enabling active spaces. Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, pp. 229-234

[73] Roman, M., Hess, C., Cerqueira, R., & Campbell, R. (2002). Gaia: A Middleware Infrastracture to Enable Active spaces. *IEEE Pervasive Computing Magazine* , pp. 74 - 83

[74] Sasse, M., & Adams, A. (1999). Taming the wolf in sheep's clothing: privacy in multimedia communications. *ACM international conference on Multimedia* Orlando: ACM Press, pp. 101-107

[75] Satyanarayanan, M. (2001). Pervasive Computing: Vision and Challenges. *IEEE Personal Communication* , pp. 10-17.

[76] Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The Case for VM-Based Cloudlets in Mobile Computing. *Pervasive computing , 8* (4).

[77] Schechter, B. (1999). Seeing the light: IBM's vision of life beyond the PC. IBM Press.

[78] Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. *Workshop On Mobile Computing Systems and Applications*. Santa Cruz: IEEE, pp. 89-101

[79] Scholtz, J. (2001). Workshop on Evaluation Methodologies for Ubiquitous Computing. Ubicom.

[80] Source, S. (2003). *Aspect Oriented Programming with Spring*. (I. 21, Producer) From http://static.springsource.org/spring/docs/2.0.x/reference/aop.html

[81] Sousa, J., & Garlan, D. (2002). Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. *3rd Working Conference on Software Architecture*, US: IEEE, pp. 102-106

[82] Sumi, H. (2006, Oct-Dec). SODA: Service-Oriented Device Architecture. *Pervasive Computing* , Volume 5, pp. 94 - 96

[83] Tanenbaun, A., & Steen, M. (2006). *Distributed Systems: principles and paradigms*. New-York: Prentice Hall.

[84] UDDI. (2000). *Universal Description, Discovery and Integration of Business for the Web*. From Universal Description, Discovery and Integration of Business for the Web: http://www.uddi.org

[85] Vaquero, L. M.-M. (2008, 12). A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun.* , pp. 50-55.

[86] W3C. (2008). *Simple Object Access Protocol*. From SOAP: http://www.w3.org/TR/soap/

[87] W3C, (2008a). *Web Services Description Language (WSDL)*. From Web Services Description Language (WSDL): http://www.w3.org/wsdl

[88] Waldo, J. (1999). The jini architecture for network-centric computing. *Communications of the ACM*, Volume 42, pp. 76-82

[89] Washington, U. o. (2002). *Portolano: An Expedition into Invisible Computing*. Retrieved 08 2008 from http://portolano.cs.washington.edu/

[90] Weiser, M. (1991). The Computer of the 21st Century. *Scientific American* , Volume 265, pp. 94-101.

[91] Weiser, M., & Brown, J. (1998). The coming age of calm technology. *Beyond calculation: The next fifty years of computing*, pp. 75-82.

[92] Wejchert, J. (2000). *The Disappearing Computer.* European Commission, Information Document, IST Call for proposals. Brussels: European Commission.

[93] Xiao, Y. (2007). Security in distributed, frid, mobile and pervasive computing. US: Auerbach Publications.

[94] Zhang X., Kunjithapatham A., Jeong S. and Gibbs S. (2011). Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing , *Mobile Networks and Applications.* Springer link, Volume 16, issue 3, pp. 270 - 284

[95] Zhou, Y., & Jiannong, C. (2007). A Middleware Support for Agent-Based Application Mobility in Pervasive Environments. *27th International Conference on Distributed Computing Systems Workshops.* IEEE, pp. 9 – 9o